UNIVERSITY OF COPENHAGEN

MASTER THESIS

Neural Networks For Univariate And Bivariate Contingent Claims

Author:
Peter Pommergård LIND

Supervisor:
Dr. David SKOVMAND

A thesis submitted in fulfillment of the requirements
for the degree of Master Thesis in Actuarial Mathematics

October 27, 2020

Declaration of Authorship

I, Peter Pommergård LIND, declare that this thesis titled, "Neural Networks For Univariate And Bivariate Contingent Claims" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"You were hired because you met expectations, you will be promoted if you can exceed them."

Saji Ijiyemi

UNIVERSITY OF COPENHAGEN

Abstract

Department of Mathematical Science
Science

Master Thesis in Actuarial Mathematics

Neural Networks For Univariate And Bivariate Contingent Claims

by Peter Pommergård LIND

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

I would first like to thank my thesis advisor Associate Professor David Skovmand of the department of Mathematical Science at University of Copenhagen. Prof. Skovmand consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to thank Joakim Pagels and Emil Petersen for providing useful discussions in the process. A special thanks go to senior lecturer Mark Laplante at University of Wisconsin Madison for be such an inspiring lecturer and discovering my interest in finance.

Finally, I must express my very profound gratitude to my parents and to my family and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Peter Pommergård Lind

Contents

List of Figures

List of Tables

List of Abbreviations

ATM	**A**t **T**he **M**oney
B-S	**B**lack-**S**choles
BM	**B**rownian **M**otion
FPT1	**F**undamental **P**ricing **T**heorem **I**
FPT2	**F**undamental **P**ricing **T**heorem **II**
GBM	**G**eometric **B**rownian **M**otion
ITM	**I**n **T**he **M**oney
LIBOR	**L**ondon **I**nter**b**ank **O**ffered **R**ate
MLPs	**M**ulti**L**ayer **P**erceptron**s**
MRT	**M**artingale **R**epresentation **T**heorem
OTM	**O**ut **T**he **M**oney
RNVF	**R**isk **N**eutral **V**aluation **F**ormula
SDE	**S**tochastic **D**ifferential **E**quation
S-F	**S**elf-**F**inancing

List of Symbols

$\mathbf{A}$	matrix notation for matrix A
$\boldsymbol{a}$	vector notation for vector a
c	European call option price
C	American Call option price
p	European put option price
P	American Put option price
K	Strike price
T	Maturity in years
σ	Volatility of asset
$S(0)$	Spot price
$S(T)$	Stock price at maturity
$S_i(t)$	i'th stock price at time t
r	Continuous compounding risk-free yearly interest rate
$V^h(t)$	Value process
X	Simple Derivative
Φ	Contract function
W_t	Weiner process under martingale measure Q (synonym brownian motion)
$\bar{W}_t$	Weiner process under probabilty measure P
p_{ij}	Correlation coefficient between asset i and j
μ_i	drift of the continuous lognormal distribution
$F(t, S(t))$	pricing function of S(t) to time t
d	number of risky assets
$\mathbb{N}$	natural numbers: $1, 2, \ldots$

For/Dedicated to/To my…

Chapter 1

Introduction

Theory of option pricing dates back to Louis Bachelier with his PhD thesis "The Theory of Speculation" in 1900, but it was much later that option theory gained significant attention. In 1973 the world's first option exchange in Chicago opened, and in the same year Fisher Black and Myron Scholes came out with the first analytical formula for the european call option (Black and Scholes, 1973), this revolutionized market practice and option pricing theory. The idea of replication was born and the financial derivatives could now be priced by rational pricing.

The Black-Scholes model for european options is still used today, but the analytical framework cannot handle more complex products such as american put options. Since that time we have seen an increasing complexity of financial products, where big investment- and banks have increased their need for financial engineerers to handle the derivative books and price derivative products. With the complexity a lot of challenges have risen in this field, where a great understanding of the products is required to handle big derivative books. Most of the existing derivatives does not have a closed form solution, so numerical methods are used to approximate the price function. To emphasize the risk of derivatives without great understanding the successful trader Warren Buffett says derivatives is "Financial weapons of mass destruction" (p. 15 (Buffett, 2002)), but on the otherhand Warren Buffett has derivatives in his portfolio. A recent example of bad management of your derivative book is AIG, where AIG needed a bailout by the US government under the recent financial crisis (McDonald and Paulson, 2015). To sum up derivatives give the trader more options either to utilize arbitrage, speculate or hedge, but without care or knowledge about your book of derivative the outcome can be disastrous.

What is the fair price for the right to buy or sell certain derivative for some predetermined exercise dates? The thesis will try to answer the question by approximating the arbitrage free pricing function, which is essential for handling a derivative book. The pricing function is essential for arbitrageurs, speculators or hedgers in the market. The aim is to give an accurate and fast method for arbitrage free pricing with neural networks, where classical methods will be used for comparision. The focus in the thesis will be on financial equity contingent claims, where the prime example will be american equity put options with one or two underlying stocks.

This thesis starts presenting the basic theory on arbitrage theory to introduce the basic modelling framework for contingent claims. With this introduction the goal is to explore the applications and numerical procedures springing from the underlying theory. The classical methods in chapter 3 such as the binomial lattice approach and least square monte carlo methods (LSM) will be applied to univariate and bivariate contingent claims, where the presentation of the classical methods are threefold. The

methods give a benchmark for the neural network approaches for option pricing in later chapters. Specially the lattice approach gives strong intuition how the optimal stopping problem can be solved, where the LSM gives a framework that is easily extended to methods for pricing multivariate contingent claims. The lattice approach makes it easy to compare values with closed form solution for european options, where some special cases for exotic european option will be presented. Deep learning theory is important for building a sound and high quality model for pricing options with neural networks, hence the basic machine learning and deep learning theory will be presented before the main chapter. The main chapter is to price univariate and bivariate claims with neural networks, where the aim is to look for methods which have high accuracy and fast computation time.

Chapter 2

Arbitrage Theory In Continuous Time Finance

Arbitrage theory in continuous time finance is a field with a lot of technical details from probability theory and stochastic calculus, where we follow the style in (Hull, 2017; Björk, 2009) to focus on intuition without going into the whelm of technicalities and proofs. The focus on this chapter will provide the basic tools and intuition for the arbitrage theory and lay the foundations for the computational finance methods. The key question is how to price derivative fairly and hedge the risk imposed by the derivative. The thesis will mainly deal with the former, where the concepts of arbitrage and replication will be important.

We start with introducing the financial markets and key concepts for building arbitrage free and complete market models (section 2.1). Then we build a framework for finding "fair" prices, i.e. finding a complete model with absense of arbitrage (section 2.2). Lastly we go into specific cases where either a closed-form solution exists or numerical methods are needed (section 2.3 and 2.4).

2.1 Financial Markets

In the financial markets there are a lot of players and different types of investments. The classical investment types are bonds and stocks, where the big players in the markets are commercial banks, investment banks, insurance companies and pension funds. Besides the classical investments types are derivatives that gives additional options for investment. A derivative or a contingent cliam is a financial instrument depending on an underlying asset, where the dependency is specified in the contract. We will focus on contingent claims with one or two underlying stocks, i.e. univariate and bivariate contingent claims, but the techniques developed can easily be extended to other types of derivatives or multivariate contingent claims. To find prices of contingent claims in modelling we restrict our financial market to d risky assets $\boldsymbol{S}(t) = (S_1(t), S_2(t), \ldots, S_d(t))$ and a bank account $S_0(t)$ as numeraire. The probability space $(\Omega, \mathcal{F}, P)$ with a filtration $\mathbb{F} = (\mathcal{F}_t)_{t\geq 0}$ is the fundamental for modelling stochastic processes describing asset prices and trading strategies, where in the thesis the probability space $(\Omega, \mathcal{F}, \mathbb{F}, P)$ will be implicit assumed. Intuitively the filtration $\mathcal{F}_t$ is the information observable to time t, where the filtration $\mathbb{F}^W$ is often used in the thesis and it is generated by the wiener processes $(W_t)_{0\leq t\leq T}$. The bank account is assumed to be a strictly positive adapted process $S_0 = (S_0(t))_{t\geq 0}$ and $S_0(0) = 1$, where the d risky assets are modelled by a $\mathbb{R}^d$ adapted stochastic proces $\boldsymbol{S} = (\boldsymbol{S}(t))_{t\geq 0}$. The risky assets are stocks where the stocks are assumed positive $S_i(t) \geq 0$ P-a.s for all i and t by financial reasons. By using the bank account as

numeraire i.e. dividing the traded asset by the bank account ($\frac{S(t)}{S_0(t)}$), this amounts to working with *zero interest*. We assume that the our financial market is frictionless.

Assumption 2.1. ***Frictionless Market:*** *We assume following institutional facts:*

- *Short positions and fractional holdings are allowed*
- *There are no bid-ask spread, i.e. selling price is equal to buying price*
- *There are no transactions costs, taxes or margin requirements of trading*
- *The market is completely liquid, i.e. it is possible to buy/sell unlimited quantities on the market. You can borrow unlimited amount from the bank by short selling*

(p. 6 (Björk, 2009))

Besides the assumptions in (Björk, 2009) we assume the market gives same uniform price for borrowing money and stocks are fixed stochastic processes exogenously and a priori given. All the assumptions are necassary not realistic in real financial markets, but it gives a reasonable approximation. The assumptions can be modified but this complicates the mathematical convience, but the financial market is the key to price derivative in arbitrage teory.

2.1.1 Contingent Claims

A contingent claim is a contract on a underlying asset or assets, where the price of the claim is contingent on the price behavior of the underlying asset. We investigate stock derivatives with different types of contracts, where we will mainly divide the derivatives into two classes.

1. Simple stock derivatives (T-claims - european options)
2. Exotic stock derivatives (e.g. American options)

The first class are simple because you can only exercise them at maturity (time T). Actually, we have a closed form solution for european options (section 2.16). The exotic derivatives are a broad class of functions on the underlying assets, where you can e.g. have an american option where the holder can exercise from inception to maturity (section 2.4) or a contract on several underlying stocks.

Definition 2.2. European Call And Put Options: A European call option is an option where the owner of the option has the option to buy the underlying asset to price K at maturity. If the owner of the option chooses to buy the underlying asset, then the option is exercised. The contract function for the european call option:

$$\Phi(S(T)) = \max\{S(T) - K, 0\} \tag{2.1}$$

The put option is the right to sell the underlying asset to price K at maturity, hence the contract function for the european put option is:

$$\Phi(S(T)) = \max\{K - S(T), 0\} \tag{2.2}$$

Where S(T) is the price of underlying asset at maturity and K is the agreed strike price.

The american option adds the feature to the european option, that you can exercise at anytime beetween inception of the contract until maturity. This feature makes the american option an optimal stopping problem and the exercise gain for the american options is called the intrinsic value (section 2.4). For the american put option the payoff function at the stopping time is the same as for the european put at maturity.

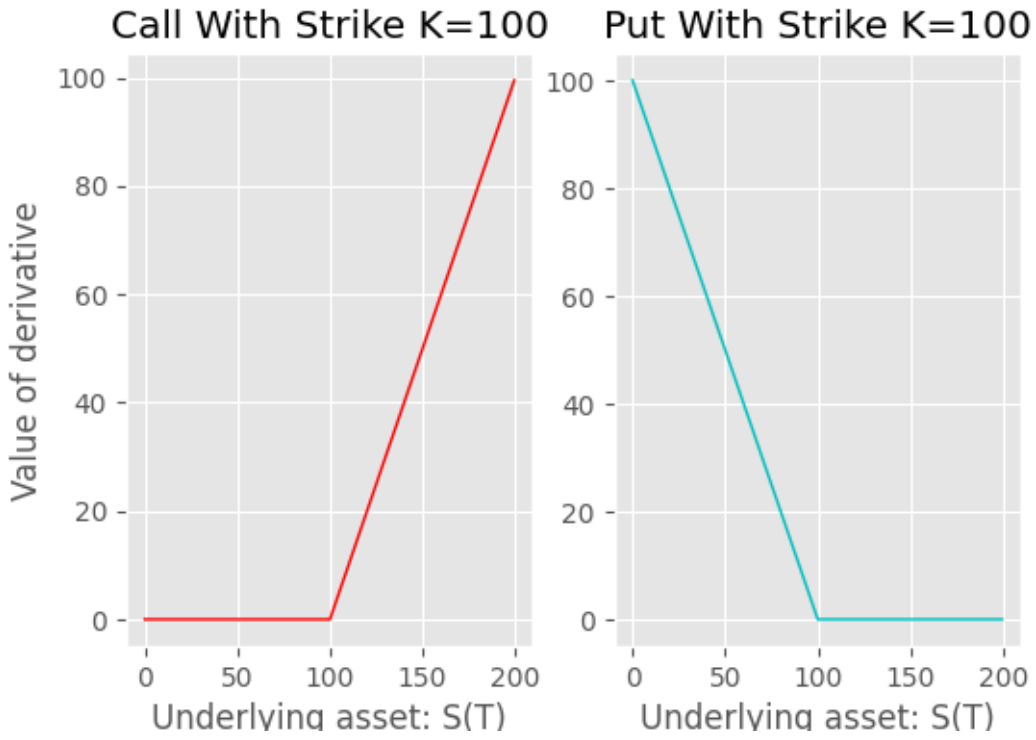

FIGURE 2.1: European options payoff at maturity with strike K=100

From the above illustration it is clear, that the owner of the option has limited downside, but the illustation does not take into account the initial price for the option. The profit and loss ($P\&L$) graph is also a common way of illustrating the payoff for an option, where you take the initial cost of buying the option into account. According to the topic for this thesis the fair price for these contracts and other exotic contracts will be investigated, where the concepts of completeness and arbitrage will be central.

2.1.2 Self-financing Portfolio (Without Consumption)

Before being able to use the concepts of arbitrage and compleness, the construction of the portfolio from the financial market model will be important. The portfolio is the number of each assets of the market the owner of the portfolio holds. The value of the portfolio for a market model with the bank account and d stocks:

$$V^h(t) = \sum_{i=0}^{d} h_i(t) S_i(t) \tag{2.3}$$

V^h is called the value process and $h_i(t)$ is number of shares of type i during the period "$[t, t+dt)$". For the definition of arbitrage (Definition 2.5) we need to restrict

ourselfes to self-financing (S-F) portfolios. A self-financing portfolio h, is a portfolio h which doesn't get any external injection of money.

Definition 2.3. Self-financing portfolio: A portfolio consisting of d+1 asset(s): h(t)=$(h_0(t), h_1(t), \ldots, h_d)$ is self-financing if:

$$dV^h(t) = \sum_{i=0}^{d} h_i(t) dS_i(t) \tag{2.4}$$

Where S_i is the $i'th$ asset in our portfolio, d+1 is the total number of assets in our market model and
$V^h(t) = \sum_{i=0}^{n} h_i(t) S_i(t)$

When dealing with discrete time finance the S-F portfolio is actually a budget restriction, this is important intuition for the continuous time version, because the continuous time version can be thought of the limit of the discrete version by letting step sizes in time tending to zero. To avoid pathological effects on the portfolio one often introduce the concept of an admissible portofolio:

Definition 2.4. A-admissible portfolio: For $a \geq 0$, a portfolio h is called a-admissible if its value process $V^h(t)$ is uniform bounded from below by -a. A portfolio h is admissible if it is a-admissible for some $a \geq 0$.

The definition of a-admissible portfolio is to avoid situations as the doubling strategy known from gambling and imposes a limit to the debt arrangement. The important takeaway is that the S-F portfolio is a portfolio where you only reallocate your assets through time within the portfolio.

2.1.3 Arbitrage

Arbitrage is the financial term for a "free lunch". An arbitage opportunity produces something out of nothing without risk, where the efficient market assumptions tells us in a well function market the "money pumps" cannot exist for long, because they would quickly be corrected by exploitation. In order to avoid making a "money machine" in our market, we want to price derivatives by not introducing arbitrage to the market.

Definition 2.5. Arbitrage: An arbitrage possibility on a financial market is an admissible self-financed portfolio h suct that

$$\begin{aligned} V^h(0) &= 0 \\ P(V^h(T) \geq 0) &= 1 \\ P(V^h(T) > 0) &> 1 \end{aligned} \tag{2.5}$$

The financial market $\boldsymbol{S}$ is called arbitrage-free if there exist no arbitrage opportunities. Sometimes $\boldsymbol{S}$ is said to satisfy (NA)
(p. 96 (Björk, 2009))

From the above definition we see that arbitrage is a natural financial requirement for a financial market model, because the investor in a arbitrage portfolio starts with 0 dollars, and without injecting any money, the investor is certain of not losing any money. In addition he has a positive probability by ending up with more than 0 at

maturity this cannot be a well function market for both buyers and sellers. To price the derivatives fair in the model, the derivative should not introduce arbitrage to the market. There are different versions of the (*NA*) definition, where for our purpose the above defition is sufficient. The *NA* is not only desirable for the market, we would also like to be able to replicate the payoff of the derivative with the other assets in the financial market model. If every derivative can be replicated in the market, then we have a complete market.

2.1.4 Complete Market And Replication

The replication argument in Black-Scholes paper (Black and Scholes, 1973) was groundbreaking in the sense that the attitude to risk was irrelavant for pricing, because by continuous trading in the underlyings of the contingent claims cash flow could be replicated. The replication argument shows that the price is unique under the assumption investors prefer more to less. Replication is also important for risk management of the derivative books, because it tells you have to risk neutralize your exposure. A replication or hedge is simply a risk neutralization action in order to minimize the overall risk. In the definition below, we define a hedge for an simple T-claim.

Definition 2.6. Hedging and completeness for T-claim: A T-claim X can be hedged, if there exist a self-financing portfolio h such that:

- $V^h(T) = X$ P-a.s.

I.e. h is an hedge portfolio for X if it is guaranteed to pay in all circumstances an amount identical to the payout of X.
The market is complete, if every derivative is hedgeable. (p. 192 (Björk, 2009))

By introducing the basic concepts for how to price fair and protect ourselves against financial risk, we will in next section focus on building the financial market model.

2.2 Multidimensional Models

There is two main method for deriving arbitrage free and complete markets. The classical approach is the delta hedging approach (Black and Scholes, 1973) and (Cox and Stephen Ross, 1979)). The more advanced mathematical approach is the martingale approach (Björk, 2009). In this section will we focus on the martingale approach and show that delta hedging approach coincides with the more general martingale theory. For the martingale approach the First and Second Fundamental Theorems of Mathematical Finance will be the key for obtaining a fair market. Besides the model assumptions will we also assumes the financial market assumptions in section 2.1.

2.2.1 Model Assumptions

Let us consider a filtered probability space $(\Omega, \mathcal{F}, P, (\mathcal{F}_t^{\bar{W}})_{t\in[0,T]})$. Note the assumption that filtration is generated from the Wiener process and we consider a finite horizon. we assume $\bar{W}_i$ is k-dimensional and $\bar{W}$ is the only random source . I.e. we assume that we are in a Wiener world, where all processes are Wiener driven. A priori we assume a market $(B(t), S_1(t), S_2(t), \ldots, S_d(t))$, where $S_i(t)_{i=1,2,\ldots,d}$ are d risky

assets and B(t) is the risk free asset. By assumptions their dynamics are given by:

$$dS(t) = D[S(t)]\alpha(t)dt + D[S(t)]\sigma(t)d\bar{W}(t) \qquad S_i(0) \in \mathbb{R}^+ \qquad (2.6)$$

$$dB(t) = r(t)B(t)dt \qquad B(0) = 1 \qquad (2.7)$$

We assume α_i, σ_{ij} and the short rate $r(t)$ are adapted processes, this condition are necassary for the stochastic integrals to be well-defined. The evolution of the stocks are described by a geometric brownian motion (GBM) which has a solution to the SDE. The randomness comes from the brownian motion (BM) in the GBM, which has wildly tracjectories. The function $t \mapsto W_t(\omega)$ from $[0, \infty)$ to $\mathbb{R}$ is continuosus, but nowhere differentiable. Furthermore has the BM nonzero quadratic variation and infinite variation, which is the reason to stochastic calculus pioneered by Itô. The BM has also well-behaved property e.g. it is a Lévy process, i.e. $W(0) = 0$ $a.s$, independent and stationary increments $W(t) - W(s)$ which is normally distributed with mean zero and variance t-s. The brownian motion in the GBM formalizes "random shocks" dW to the stock return with volatility σ and α is the drift. Figure 2.2 illustrates three approximations to sample paths of the stocks with GBM assumption with initial value $S_i(0) = 36$.

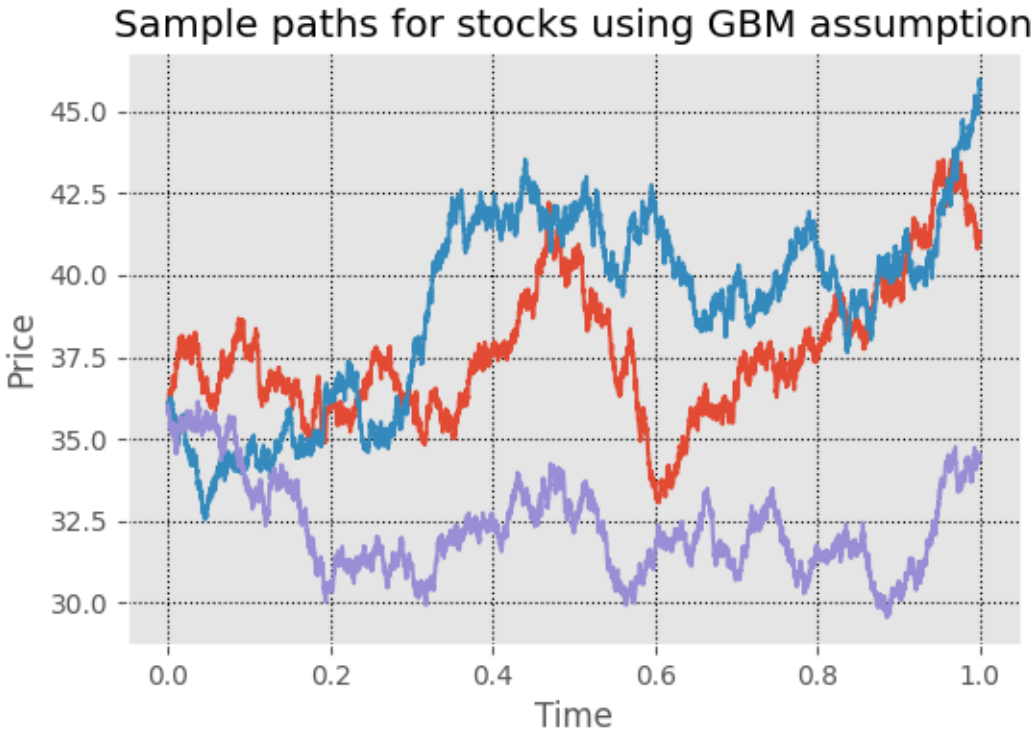

FIGURE 2.2: Three sample paths for stocks under GBM assumptions, where the spot is \$36, σ=0.2 and α=0.06

The tool for handling BM is stochastic calculus in continuous time, because the standard calculus will not work for the wildly behaved BM. In the representation of the GBM, we used vector and matrix notation for the GBM process. The stock vector is d dimensional and the wiener process vector is k dimensional. The volatility matrix is given by $\sigma(t) = \{\sigma_{ij}(t)\}_{i=1,\dots,d,j=1,\dots,k}$ and the local mean of rate of return

vector is $\alpha(t) = (\alpha_1(t), \alpha_2(t), \ldots, \alpha_d(t))^T$. D(x) denotes a diagonal matrix with vector x as its diagonal and the Wiener processes instantaneous correletion matrix ρ is $Cov(dW_i(t)dW_j(t)) = \rho_{ij}dt$.

2.2.2 Arbitrage Free Model

The first problem we are faced with in arbitrage theory is to create a model with no arbitrage opportunities. The First Fundamental Theorem tells us how to not introduce arbitrage to our market model.

Theorem 2.7. ***First Fundamental Pricing Theorem of Mathematical Finance(FFT1):*** *The market model is free of arbitrage if and only if there exist a equivalent martingale measure, i.e. a measure $Q \sim P$ s.t. the processes:*

$$\frac{S_0(t)}{S_0(t)}, \frac{S_1(t)}{S_0(t)}, \ldots, \frac{S_d(t)}{S_0(t)}$$

are (local)martingales under Q. (see p. 154 (Björk, 2009))

The processes are martingales is natural, because a martingale is mathmatical formulation that the expected value of the discounted value coincides with the known spot value. In gambling a martingale resembles a "fair" game, which is the same idea for FFT1. By choosing the bank account $B(t)$ as numeraire it follows from the FFT1.

Proposition 2.8. *We assume that $B(t) = S_0(t)$ is our numeraire and all the processes are Weiner driven, then a equivalent measure $Q \sim P$ is martingale measure if and only if all assets $(B(t), S_1(t), \ldots, S_d(t))$ have the short rate as their local rates of return, i.e.*

$$dS_i(t) = S_i(t)r(t)dt + S_i(t)\sigma_i(t)dW^Q(t) \tag{2.8}$$

(see p. 154 (Björk, 2009))

So to not introduce arbitrage to the model for the financial market, we need to ensure the Q-dynamics of S is:

$$dS(t) = D[S(t)]r(t)dt + D[S(t)]\sigma(t)dW(t) \tag{2.9}$$

The tool to obtain the dynamics in eq. (2.9) is Girsanov theorem (see A.3). Girsanov theorem is a continuous measure transformation, where in our model we want to transform the dynamics given with the objective probability measure P to an equivalent martingale measure Q (i.e. the martingale measure chosen by the market). By suitable chooses of the likelihood process L and setting $dQ = L(T)dP$, then with Girsanov theorem the transformed process is still a brownian motion:

$$d\bar{W}(t) = \phi(t)dt + dW(t)$$

When applying to eq. (2.6):

$$dS(t) = D[S(t)](\alpha(t) + \sigma(t)\phi(t))dt + D[S(t)]\sigma(t)dW(t)$$

Going back to the FFT1 and the proposition, we know that Q is martingale measure if and only if:

$$\boldsymbol{\alpha}(t) + \boldsymbol{\sigma}(t)\boldsymbol{\phi}(t) = \mathbf{r}(t) \quad \textit{holds with probability } 1 \textit{ for each } t \tag{2.10}$$

We disregard pathological models when doing so the term generically arbitrage free will be used.

Definition 2.9. Generically arbitrage free: The model in this section is said to be generically arbitrage free if it is arbitrage free for every (sufficiently integrable) choice of $\boldsymbol{\alpha}$. (p. 198 (Björk, 2009))

Furthermore we assume enough integrability and we have the following useful result:

Proposition 2.10. *Disregarding integrability problems the model is generically arbitrage free if and only if, for each $t \leq T$ and P-a.s. the mapping: $\boldsymbol{\sigma}(t) : \mathbb{R}^k \to \mathbb{R}^n$ is surjective, i.e. if and only if the volatility matrix $\boldsymbol{\sigma}(t)$ has rank n. (see p. 198 (Björk, 2009))*

We note that in order not to have arbitrage in our model, we need $k \geq n$, i.e. have at least as many random sources as number of risky assets.

2.2.3 Complete model

Second Fundamental Pricing Theorem is key to obtain a complete market model, i.e. a market model where every claim can be hedged.

Theorem 2.11. ***Second Fundamental Pricing Theorem of Mathematical Finance(FFT2):*** *Assuming absence of arbitrage, the market model is complete if and only if the martingale measure Q is unique. (see p. 155 (Björk, 2009))*

In our Wiener world we have a unique martingale measure if eq. 2.10 has a unique solution, where from the proof of the proposition 2.12 is clear why the Wiener world assumption is required. If we had more random sources e.g. a poisson process, than there is no guarentee that the equivalent measure transformation is of the Girsanov type above.

Proposition 2.12. *Assume that the model is generically arbitrage free and that the filtration is defined by:*

$$\mathcal{F}_t = \mathcal{F}_t^{\bar{W}} \quad t \in [0, T]$$

Then disregarding integrability problems, the model is complete if and only if k=n and the volatility matrix $\sigma(t)$ is invertible P-a.s. for each $t \leq T$

Proof. The proof is based on martingale representation theorem A.2 (MRT) and converse of Girsanov theorem A.4 which uses MRT, hence the assumption about the only randomness comes from the wiener process. By the two theorems we know every equivalent meassure transformation is obtained by Girsanov theorem of the above type. Hence the martingale measure is unique if anf only if the solution to (2.10) is unique (p. 200 (Björk, 2009)). ■

Intuitively we need one independent traded assets excluding the bank account for every source of randomness (Meta-theorem 8.3.1 (Björk, 2009)).

2.2.4 Pricing and connection to classical approach

The pricing formula for arbitrage free market model is the risk neutral valuation formula:

Proposition 2.13. ***Risk Neutral Valuation Formula*** *To avoid arbitrage, $\mathcal{X}$ must be priced according to the formula:*

$$\Pi(t;\mathcal{X}) = S_0(t)E^Q[\frac{\mathcal{X}}{S_0(T)}|\mathcal{F}_t] \tag{2.11}$$

Note if we choose our numeraire $S_0(t) = B(t)$ then

$$\Pi(t;\mathcal{X}) = E^Q[\exp(-\int_t^T r(s)ds)\mathcal{X}|\mathcal{F}_t] \tag{2.12}$$

(see p. 155 (Björk, 2009))

Proposition 2.13 will raise the question if there is more than one fair price for the derivative. The answer is found in FTT2, the market is complete if and only if the measure Q is unique. Intuitively it means that if you can replicate the derivative with a S-F portfolio, then the risk free position should only earn risk free interest rate.

The classical approach in (Black and Scholes, 1973) to arbitrage free and complete market models is based on a Markovian model assumption. For the model to have markovian property, we assume k=n and the probability space is $(\Omega, \mathcal{F}, P, \mathcal{F}_t^{\bar{W}_t})$. Furthermore we assume $\boldsymbol{\alpha}$ and σ are deterministic and constant over time. σ is also assumed invertible. Under these more restrictive assumptions the risk neutral valuation formula for a simple T-claim is given by the pricing function:

$$F(t, S(t)) = \exp(-r(T-t))E^Q[\mathcal{X}|S(t)] \tag{2.13}$$

The Markov property implies that the price only depend on the current state of $\boldsymbol{S}$, and then applying Kolmogorov backward equation on eg. 2.13, we obtain the BS-PDE for the pricing function F(t,S(t))=$\Pi(t;\mathcal{X})$.

Theorem 2.14. ***Black Scholes PDE:*** *Consider the contract $\mathcal{X} = \Phi(\boldsymbol{S}(T))$. In order not to introduce arbitrage to the market, the pricing function $F(t,s)$ must solve the boundary value problem.*

$$\begin{aligned} F_t(t,s) + \sum_{i=1}^{n} rs_i F_i(t,s) + \frac{1}{2}tr\{\sigma^* D[S]F_{ss}D[S]\sigma\} - rF(t,s) = 0 \\ F(T,s) = \Phi(s) \end{aligned} \tag{2.14}$$

(see p. 203 (Björk, 2009))

2.3 Classical Black-Scholes Formulas

We will not do the classical delta hedging approach in (Black and Scholes, 1973). Instead we use the general multidimensional martingale approach to derive the essential formulas for pricing. To derive a closed-form solution to the European call and put option, we concentrate at a special case of the multidimensional framework, where we only have the risk free asset and one risky asset in the financial market model. We further restrict ourselves to:

Assumption 2.15. ***Black-Scholes assumptions****: We assume following ideal conditions in addition to (2.1):*

- *The short-term interest rate r, volatility σ and the drift α are constant.*

- *The stock pays no dividends or other distributions.*
- *The option is a simple option ("European").*

(p. 640 (Black and Scholes, 1973))

We assume the underlying stock and the bank account have differentials:

$$dS(t) = S(t) \cdot \alpha dt + S(t)\sigma d\bar{W}(t) \qquad S(0) \in \mathbb{R}^+ \tag{2.15}$$
$$dB(t) = rB(t)dt \qquad B(0) = 1 \tag{2.16}$$

By Itô's lemma (lemma A.1) for one dimensional process the solution to the differentials above is:

$$S(t) = S(0) \cdot \exp\left((\alpha - \frac{1}{2}\sigma^2)t + \sigma W(t)\right) \tag{2.17}$$
$$B(t) = \exp(r \cdot t) \tag{2.18}$$

Where α is the local mean rate of return and σ is the volatility of S are constants. By above assumptions we are in a Markovian model, and we know that the Black Scholes PDE in this setup (see eq. 2.14) from the risk neutral valuation formula. The solution of the SDE of S under Q dynamics is:

$$S(t) = S(0) \cdot \exp\left((r - \frac{1}{2}\sigma^2)t + \sigma W(t)\right) \tag{2.19}$$

By eq. (2.19) we see that the Black-Scholes model assumes that the stock price change continuously in a way that produces a lognormal distribution for the price at any future time.

By the simple option assumption the Kolmogorov backward equation for the RNVF gives the BS-PDE. The PDE can then be solved analytically and then we arrive at a closed form solution for european call and put options. The european call and put option can also be derived directly with the RNVF.

Proposition 2.16. ***Black-Scholes formula for call option:*** *The price of a European call option with strike K and maturity T is given by the formula $\Pi(t) = F(t, S(t))$, where*

$$F(t,s) = c(t,s) = s \cdot N(d_1(t,s)) - e^{-r(T-t)} \cdot K \cdot N(d_2(t,s))$$

N is the cumulative distribution function of a standard normal distribution $\mathcal{N}(0,1)$ and

$$d_1(t,s) = \frac{1}{\sigma \cdot \sqrt{T-t}} \cdot \left(\ln(\frac{s}{K}) + (r + \frac{1}{2}\sigma^2)(T-t)\right)$$
$$d_2(t,s) = d_1(s,t) - \sigma\sqrt{T-t}$$

(see p. 105 (Björk, 2009))

We provided only the price for the european call option, but the european put price can easily be obtained by applying the put-call-parity for european options.

Proposition 2.17. ***Put-call parity:*** *Assume the call and put option has same strike price and time to maturity.*

$$p(t,s) = K \cdot \exp(-r(T-t)) + c(t,s) - s$$

(see p. 126 (Björk, 2009))

The aim for this thesis is to price american put options, but the european option provide a reference price in a closed form format. The put-call-parity holds only for european options, where for the american option there is a bound on the difference in price:

$$S_0 - K \leq C - P \leq S_0 - K \cdot e^{-rT}$$

The above formula for the European call option is actually the same for an American call option, but is not true for an American put option or for call options with underlying stock paying dividends. The result for the American call option was shown by Merton (Merton, 1973), that the intrinsic value is never greater than the worth of the option given by the risk-neutral valuation formula. In section 2.4 we will show a martingale approach to prove the value of a European and American call coincides when the underlying is a non-dividend paying stock.

2.4 American Options And Optimal Stopping

The American options adds additional complexity to the pricing problem, because compared to the European option the American option can be exercised at any time from inception to maturity. The exercise value at time t is also called the intrinsic value of the option. This section is inspired by (Björk, 2009; Shiryaev and Peskir, 2006; Elliott and Kopp, 1999) where (Shiryaev and Peskir, 2006) is specialized to optimal stopping problems, where the two other references gives the fundamental mathematics for option and arbitrage theory in general.

We still assume a diffusion setting that the underlying stochastic process for the stock behaves under the risk neutral measure as a GBM.

$$S_i(t) = S_i(0) \cdot \exp\left(\sum_{j=1}^{d} (\sigma_{i,j} W_j(t) - \frac{1}{2}\sigma_{i,j}^2 t) + rt \right) \quad for\ i = 1, \ldots, d$$

Where the Wiener processes are independent and its domain in $\mathbb{R}^+$. The exercise feature of the american option raises the problem of rationally to find the optimal stopping time to maximize profit. The value of the option is given by exercising the option at the optimal stopping time, hence it is a optimal stopping problem. We will assume a finite horizon $T \in \mathbb{R}_*^+$ throughout the thesis, because all the derivative will be priced in a finite timeframe. Let the gain function $G : \mathbb{R} \to \mathbb{R}$ be a measurable function satisfying:

$$E_s[\sup_{0 \leq t \leq T} |G(S(t))|] < \infty \tag{2.20}$$

where S is the underlying stochastic process. If the integrability condition is satisfied on a finite interval $[0, T]$ (equation (2.20)) then the optimal stopping problem for gain function G and $s \in \mathbb{R}$ is well defined. We assumed that the underlying stochastic S(t) process is time-homogeneous, but the assumption can be relaxed. If S(t) is a time-inhomegenous we can extend the underlying process S(t) by time albeit increasing the underlying process dimension. We define the optimal value process in terms of the gain process.

Definition 2.18. For fixed $(t,x) \in [0,T] \times \mathbb{R}$, and each stopping time τ with $\tau \geq t$ the optimal value function $V(t,x)$ is defined by

$$V(t,x) = \sup_{\tau \in \mathcal{T}^T} E_{t,x}[G(S(\tau))] \tag{2.21}$$

A stopping time which realizes supremum for V is called optimal and be denoted $\hat{\tau}$. (see page 341 (Björk, 2009))

The solution to the optimal stopping problem $\hat{\tau}$ is where supremum is attained and the price is then $V(t,x)$ for $(t,x) \in [0,T] \times \mathbb{R}$. The supremum is taken over all stopping times with respect to the natural filtration $\mathcal{F}_t$ belonging in the class of stopping times:

$$\mathcal{T}_0^T = \mathcal{T}^T = \{\tau : 0 \leq \tau \leq T\}$$

The definition of a stopping time τ can be seen in definition A.5, where the intuition is that the stopping time is a random time, such that we know at time t wheater to stop. To solve the optimal stopping problem some trivial solutions are immediate by martingale properties:

Proposition 2.19. *The following hold:*

- *If $G(S(t))$ is a submartingale, then it is not optimal to stop at all and $\tau^* = T$*
- *If $G(S(t))$ is a martingale, then all stopping times $\tau \in [0,T]$ are optimal*
- *If $G(S(t))$ is a supermartingale, then it is optimal to stop immediately. i.e. $\tau^* = 0$*

(p. 330 (Björk, 2009))

Examples of gain functions could be the american call and put options:

$$C(t,x) = \sup_{\tau \in \mathcal{T}_t^T} E^Q[\exp(-r(\tau - t))(S(\tau) - K)^+ | S(t) = x] \quad \textit{for } t \in [0,T] \textit{ and } x \in \mathbb{R}^+ \tag{2.22}$$

$$P(t,x) = \sup_{\tau \in \mathcal{T}_t^T} E^Q[\exp(-r(\tau - t))(K - S(\tau))^+ | S(t) = x] \quad \textit{for } t \in [0,T] \textit{ and } x \in \mathbb{R}^+ \tag{2.23}$$

2.4.1 American Call Without Dividends

The American call options is a special case, because the optimal stopping time is always at the options maturity. With martingale machinery it means the value-process is a submartingale which implies that $\hat{\tau} = T$ (proposition 2.19). Remember the optimal stopping problem for an american call option:

$$C(t,x) = \sup_{\tau \in \mathcal{T}_0^{T-t}} E_{t,x}^Q[\exp(-r\tau)(S(t+\tau) - K)^+]$$

Looking at the gain function:

$$\exp(-rt)(S(t) - K)^+ = (\exp(-rt)S_t - \exp(-rt)K)^+$$

Recall that the discounted price process $\exp(-r \cdot t) \cdot S_t$ is a Q-martingale and $\exp(-r \cdot t) \cdot K$ is a deterministic decreasing function in t if $r > 0$. Furthermore the function $x \mapsto (x)^+$ is convex, hence the gain function is a Q-submartingale. The last result used is that a convex and increasing function on a submartingale is still a submartingale, hence the optimal stopping time is $\hat{\tau} = T$ if $r > 0$.

2.4.2 American Put

The arbitrage-free price for an american put at time t:

$$P(x,t) = \sup_{\tau \in \mathcal{T}_t^T} E^Q[\exp(-r(\tau - t))(K - S(\tau))^+ | S(t) = x] \quad for\ t \in [0,T]\ and\ x \in \mathbb{R}^+ \tag{2.24}$$

For the american put we need computational methods, because the american and european value does not coincides like for the call option.

Proposition 2.20. *Consider european and american put options with same maturity T and strike K. If the risk free rate $r \in \mathbb{R}_*^+$, then for any $t < T$*

$$p(x,t) < P(x,t) \tag{2.25}$$

Proof. WLOG[1] we assume that t=0. Define the stopping time

$$\tau = min\{t \geq 0 : S(t) \leq K(1 - \exp(-r \cdot (T - t)))\}$$

We consider a exercise strategy $min\{\tau, T\}$, where the strategy is not necassarilly optimal. We consider two events:

1) $\tau < T$

2) $\tau \geq T$

The first case is to exercise at time τ when $S(\tau) \leq K(1 - \exp(-r \cdot (T - t)))$. Here the payoff by exercing will be at least $K \exp(-r \cdot (T - \tau))$. The cashflow received is then invested into the bank account at time τ. At maturity the strategy gives the holder of the put a payoff K, where the european contract with strike K will pay less, because the stock price at maturity will be $S(T) > 0$ *a.s.* The second case is trivial, because the european and american put will give the same payoff.
Since the first case has positive probability, the american put has higher discounted expected payoff by following above strategy regardless of the spot price for the stock. ■

The above proposition shows the optimal stopping strategy is not always to hold the option to maturity, hence the theory of optimal stopping is important for pricing of american put options. The american put has consequently no closed form solution, but we can search for a lower exercise boundary b(t) such that the holder of the option should exercise when:

$$S(\tau) \leq b(\tau) \quad \tau \leq T$$

The continuation C and stopping set $\bar{D}$ are given for an american put.

$$C = \{(t,x) \in [0,T) \times (0,\infty) : V(t,x) > G(x)\} \tag{2.26}$$
$$\bar{D} = \{(t,x) \in [0,T) \times (0,\infty) : V(t,x) = G(x)\} \tag{2.27}$$

Hence the first optimal stopping time after time t for the american put is

$$\hat{\tau} = \inf\{u \in [t,T] : P(S(u),u) = (K - S(u))^+\} \tag{2.28}$$

[1] Without loss of generality

2.4.3 Discrete time valuation

To solve the optimal stopping problem numerical methods are required for the american put option, hence the first step is to discretize time. In chapter 3 will we show two approaches to price an american put option, where both are based on calculating the conditional expectation of the continuation value. Suppose the probability space $(\Omega, \mathcal{F}, P)$ is equipped with the natural discrete filtration $(\mathcal{F}_{t_n})_{n=0,1,\dots,N}$ modeling a financial market. By discretization of time we are actually looking at a bermuda option, but for sufficient small time steps the bermuda option approximate the american option well. The tenor structure is that the time to maturity is divided into a grid of N+1 equidistant points in time $0 = t_0 \leq t_1 \leq t_2, \cdots \leq t_N = T$, where $\Delta t_n = t_n - t_{n-1} = T/N$ for each $n = 1, \dots, N$. A bermuda option incepted at time t_0 has $\mathcal{T}(t_1, t_2, \dots, t_N)$ exercise dates or decision points, where the option holder chooses to exercise or keep the option alive. The underlying process is assumed to be markovian with state variables $(S(t_n))_{n=0,1,\dots,N}$ recording all necessary information about relevant financial variables adapted to the natural filtration in order to solve the optimal stopping problem with the dynamic programming principle. Futhermore is the gain process adapted to the filtration and square integrable $E[\max_{0\leq n\leq N} |G(S(t_n))|^2] < \infty$, which is necassary to use regression on a finite set of functions to approximate the conditional expectation (section 3.3).

The optimal stopping problem in discrete time is to find

$$\sup_{\tau \in \mathcal{T}(0,\dots,T)} E^Q[G(S(\tau))] \tag{2.29}$$

For using the programmable dynamic programming principle the Snell envelope $(U(t_n))_{n=0,1,\dots,N}$ (definition A.7) of the gain function $G(S(t_n))_{n=0,\dots,N}$ is useful and defined by

$$U(t_n) = \operatorname*{ess\,sup}_{\tau \in \mathcal{T}(n,\dots,N)} E^Q[G(S(\tau_{t_n}))|\mathcal{F}_{t_n}] \quad n = 0, 1, \dots, N$$

The snell envelope $U(t_n)$ is the smallest supermartingale of the gain process $\{G(S(t_n))\}$, i.e. the smallest supermartingale dominating the gain process. Using the Snell envelope theory the optimal stopping problem can be solved with the dynamic programming principle.

$$\begin{cases} U(T) = G(S(t_N)) \\ U(t_n) = max\{G(S(t_n)), E^Q[U(t_{n+1})|\mathcal{F}_{t_n}])\} \quad for\ n = 0, \dots, N-1 \end{cases} \tag{2.30}$$

The equation (2.30) known as value iteration indicates that the holder should exercise the first time $G(S(t_n)) > E^Q[U(t_{n+1})|\mathcal{F}_{t_n}])$ in order to maximize payoff from the option, hence we get the first optimal stopping time. Note that an optimal stopping time will always exist in discrete time when $T < \infty$. There is though no guarantee that the stopping time is unique. The value iteration gives the optimal value process of the gain process $G(S(t))$ (theorem A.8). So

$$U(t_n) = E^Q[G(S(\tau_{t_n})|\mathcal{F}_{t_n}]$$

with

$$\tau_{t_n} = \min\{k \geq n : U(t_k) = G(S(t_k))\}$$

and

$$E^Q[U(0)] = \sup_{\tau \in \mathcal{T}(0,\dots,T)} E^Q[G(S(\tau))] = E^Q[G(S(\hat{\tau}))]$$

The optimal stopping problem can also be solved in terms of stopping times instead of using the value process. This alternative dynamic programming principle equation is called policy iteration.

$$\begin{cases} \tau_{t_N} = t_N \\ \tau_{t_n} = t_n \cdot 1_{\{G(S(t_n)) \geq E^Q[G(\tau_{t_{n+1}})|\mathcal{F}_{t_n}])\}} + \tau_{t_{n+1}} \cdot 1_{\{G(S(t_n)) < E^Q[G(\tau_{t_{n+1}})|\mathcal{F}_{t_n}])\}} \quad for\ n = 0, \dots, N-1 \end{cases} \tag{2.31}$$

The first approach, the binomial model, uses the idea of dynamic programming, where the orginal problem is solved iterative by recursive subproblems. The underlying stochastic process is approximated by a discrete binomial recombining lattice, where the iterative scheme makes it easy to calculate both european and american option prices. The american put option can be solved recursively with dynamic programming principle with value iteration:

$$\begin{cases} P(t_i) = max\{(K - S(t_i))^+, \exp(-r \cdot \Delta t) E^Q[P(t_{i+1})|\mathcal{F}_{t_i}])\} \quad for\ i = 0, \dots, N-1 \\ P(t_N) = (K - S(t_N))^+ \end{cases} \tag{2.32}$$

The second approach, Least Square Monte Carlo Method (LSM) model, is to combine dynamic programming and monte carlo simulation, but this method uses the policy iteration principle instead of the value iteration. The method uses regression to calculate the expected continuation value of simulated paths instead of discretization of the underlying stochastic process. By using regression the method overcome the challenge with pure simulation method. In order to find a optimal exercise strategy a series of conditional expectations need to be calculated which are computational expensive to calculate using pure simulation methods. Both methods will be explained in details in chapter 3.

Chapter 3

Classical numerical results and Benchmarks

In last section we saw that the american put was an example of an option that required numerical procedures to be priced fair. The American put is far from the only example of a derivative without a closed-form solution. We will look at two classical valuation algorithms for pricing american options in computational finance the Cox-Ross-Rubinstein (CRR) binomial model (Cox and Stephen Ross, 1979) and the Least Square Monte Carlo (LSM (Longstaff and Schwartz, 2001)). The binomial model is an example of a strategy to approximate the american option by discretization of the underlying risky asset(s) and the LSM is a method to simulate the underlying risky asset(s). Another popular choice is to solve the free boundary problem with finite difference methods, but we chose to focus on the two other numerical procedures. The chapter will also investigate valuing exotic multivariate contingent cliams. We extend the binomial pricing model ((Ekvall, 1996; Boyle, Evnine, and Gibbs, 1989)) and LSM to multivariate contingent claims and provide some closed form solutions for exotic european options ((Johnson, 1987; Ouwehand, 2006)). We wrap up the chapter with numerical findings. Therefore the chapter have two purposes to gain insight into valuation for exotic options and provide some benchmarks for the Neural Network in the coming chapters.

3.1 Cox Ross Rubenstein Model

The classical binomial model pricing formula or the CRR model presented in this section is inspired by (Cox and Stephen Ross, 1979; Hull, 2017; Björk, 2009). The model will be used for pricing an american put option with 1 underlying stock and to build the foundation for pricing bivariate contingent claims with the binomial model (Boyle, Evnine, and Gibbs, 1989). The CRR model provides an intuitive and easy implementable model for valuing american and european options. The binomial model comes handy, when no analytical model exists e.g. for an american put option. The Binomial model also has its limitations, because for computational reasons it is not suited for valuing path dependent options or options with several underlying factors. The key difference on the binomial model and the simulation approach is that the binomial model discretize the underlying stochastic process(es) in time.

Assume the same market and Black-Scholes assumptions as in chapter 2, but the underlying stochastic process will be assumed to follow a multiplicative binomial process over discrete periods. We work with the financial market $(\Omega, \mathcal{F}, \mathbb{F}, P, S_0, S_1)$,

where the filtration is generated by $\mathbb{F} = \sigma(\mathcal{F}_{t_n})_{n=0,1,\ldots,N}$ and the sigma algebra is chosen to be $\mathcal{F} = \mathcal{F}_{t_N}$. It is well known from discrete arbitrage teory, that the binomial market model with two assets, where $u > 1 + r > d > 0$ is a complete and arbitrage free model. The u,d and r describes the evolution of the discrete stochastic process for the stock and the free interest rate on the bank account.

$$S_0(t_n) = S_0(t_{n-1}) \cdot \exp(\Delta t \cdot r) \quad \textit{where } S_0(0) = 1 \textit{ and } n = 0, \ldots, N$$
$$S_1(t_n) = S_1(0) \prod_{j=1}^{n} Y_j \quad \textit{where } Y_1, Y_2, \ldots, Y_N \textit{ are i.i.d., } S_1(0) > 0$$

Note that the interest rate is continuously compounded for computational convience and the equidistant time step is $\Delta t = T/N$ (section 2.4.3 for notation). We assume that

$$Y_i = \begin{cases} u & \textit{with probability } p \\ d & \textit{with probability } (1-p) \end{cases}$$

Below a illustration of a discrete multiplicative binomial process with two time steps, where the lattice recombines.

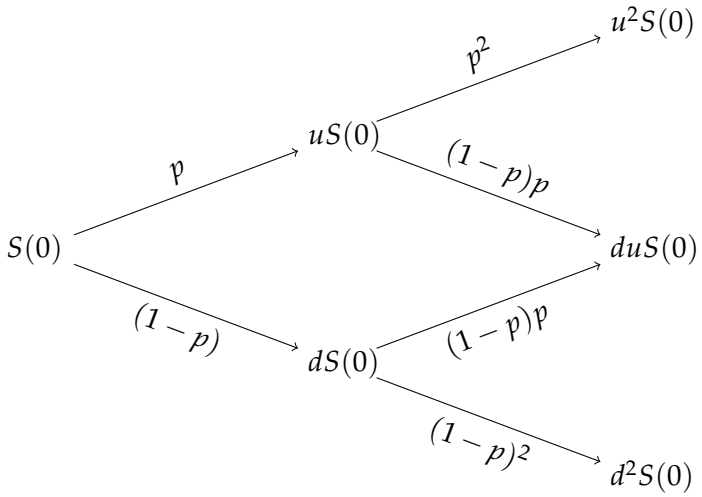

FIGURE 3.1: Price dynamics of binomial model with one underlying risky asset with N=2, $S(0)$ is spot value, p objective probability measure, u and d is realizations of the stochastic variable Y_i

By construting the discrete process for the stock it is easy to find the equivalent martingale measure.

Definition 3.1. Assume there exists a risk free asset. A probability measure Q is called a martingale measure if the following condition holds

$$s = \exp(-r\Delta t) \cdot E^Q[S(t + \Delta t)|S(t) = s] \tag{3.1}$$

Where Δt is a single time-step (p. 18 (Björk, 2019)).

By using definition 3.1 we find the risk neutral measure to be:

$$q = \frac{e^{r\Delta t} - d}{u - d}$$

The martingale measure q is unique in the binomial model, because the model is complete. By above assumption about $u > 1 + r > d > 0$ the binomial model with the market $(S_0(t_n), S_1(t_n))$ is a arbitrage free and complete model, hence to the general pricing formula is given by the risk neutral valuation formula.

Theorem 3.2. ***Risk-neutral valuation formula (RVNF) in discrete time for T-claim*** *The arbitrage free price at t=0 of a T-claim X is given by*

$$\Pi(0;X) = \exp(-r\Delta t \cdot N) \cdot E^Q[X] \tag{3.2}$$

$$= \exp(-r\Delta t \cdot N) \cdot \sum_{n=0}^{N} \binom{N}{n} q^n (1-q)^{N-n} \Phi(s u^n d^{N-n}) \tag{3.3}$$

here Q denotes the martingale measure, $\Pi(0;X)$ *is the price of X to time 0 and* Δt *is a single time step (p. 25 (Björk, 2019)).*

By above formula the binomial model gives a simple mathematical framework for pricing european options, but the model can easily be extended to american options. American put options for the binomial will be solve with the dynamic programming approach, because the binomial model gives a recursive formula where the one-step transition probabilities are the same in each node.

$$\begin{cases} P(t_i) = max\{(K - S(t_i))^+, \exp(-r \cdot \Delta t) E^Q[P(t_{i+1})|S(t_i)])\} & for\ i = 0, \dots, N-1 \\ P(t_N) = (K - S(t_N))^+ \end{cases} \tag{3.4}$$

The idea is to at each decision point either exercise to gain the intrinsic value or hold the option alive for another period. The value of keeping the option alive is called the expected continuation value and it is given by the risk neutral valuation formula. The dynamic choice to exercise or to keep the option alive gives a exercise barrier b (see section 2.4.2).

So to value an American put option, we lay out all the possible path of the stock based on $S_1(0), \sigma$ and T. These paths construct the tree, then for valuation we work backwards in the tree starting at maturity. Figure 3.2 is an example of a constructed tree, where the value of the option is also included by color. The decision to exercise is marked with a triangle where the decision points where is optimal to keep the option alive is marked with circles.

To construct the tree we need to specify the number of equidistant time-steps Δt ($\Delta t = \frac{T}{N}$ *where* $N = No.\ of\ steps$) for the tree, where for each step we add another possible value for the stock. We only add 1 more possibility for each time-step because the tree recombines [1]. The d and u is chosen such that they match volatility.

$$u = \exp(\sigma\sqrt{\Delta t}) \quad d = \exp(-\sigma\sqrt{\Delta t})$$

For valuing an American put option, we value the exercise value at maturity (time T) for all possible outcomes for the price process at maturity. Then we use backward induction/dynamic programming where we compare the intrinsic value with the expected continuation value (see (3.4)), where we choose the maximum of these two. The comparison will be applied for every node in each decision point $(t_n)_{n=0,1,\dots,N-1}$

[1] $(1+n)$ possible stock values after n steps

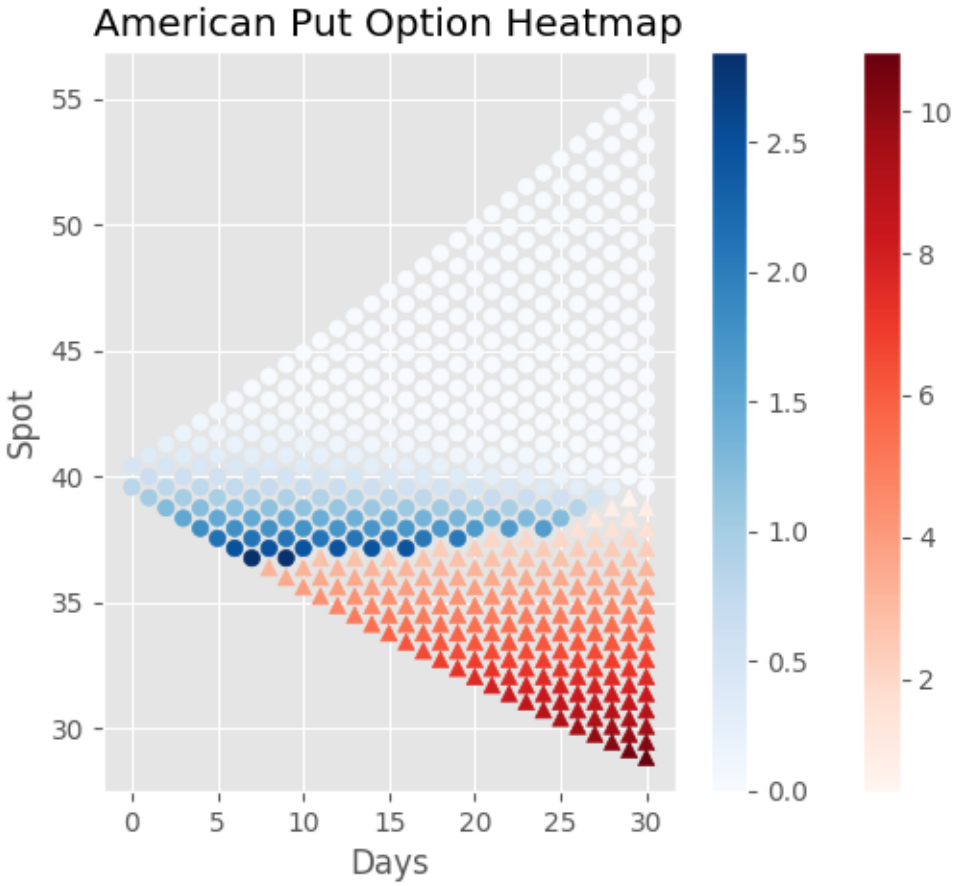

FIGURE 3.2: A valuation tree of an american put option price based on the binomial model, where the color indicate the value and the dots are marking the continuation nodes. The parameters are S(0)=40, N=30, $\Delta t = 1$ day, K=40 and u=1.0106

and all the way back in time to the initialization date. By this procedure we get present value of the American option. One design decision is to choose number of time-steps considering a trade-off between computational efficientcy and accurary. Figure 3.3 illustrates that around 40 steps the option value stabilizes for an option with 1 year to maturity. The precision for the algorithm increases with the number of steps and the option value stabilizes for increasing number of steps (see Figure 3.3).

We have seen the central concepts arbitrage and completeness from continuous time in action for the discrete time setup. The paper (Cox and Stephen Ross, 1979) which introduced the binomial model to option pricing came after the Black-Scholes model described in section 2 (Black and Scholes, 1973). The main reason for developing a model in discrete time is that the discrete time approach gives a simplified model in terms of the mathematics and highlights the essential concepts in arbitrage theory. You can argue that the simpler mathematics in this model makes the binomial model more instructive and clear. Besides being easier to understand for non-mathematician it works nicely with other options than the european options

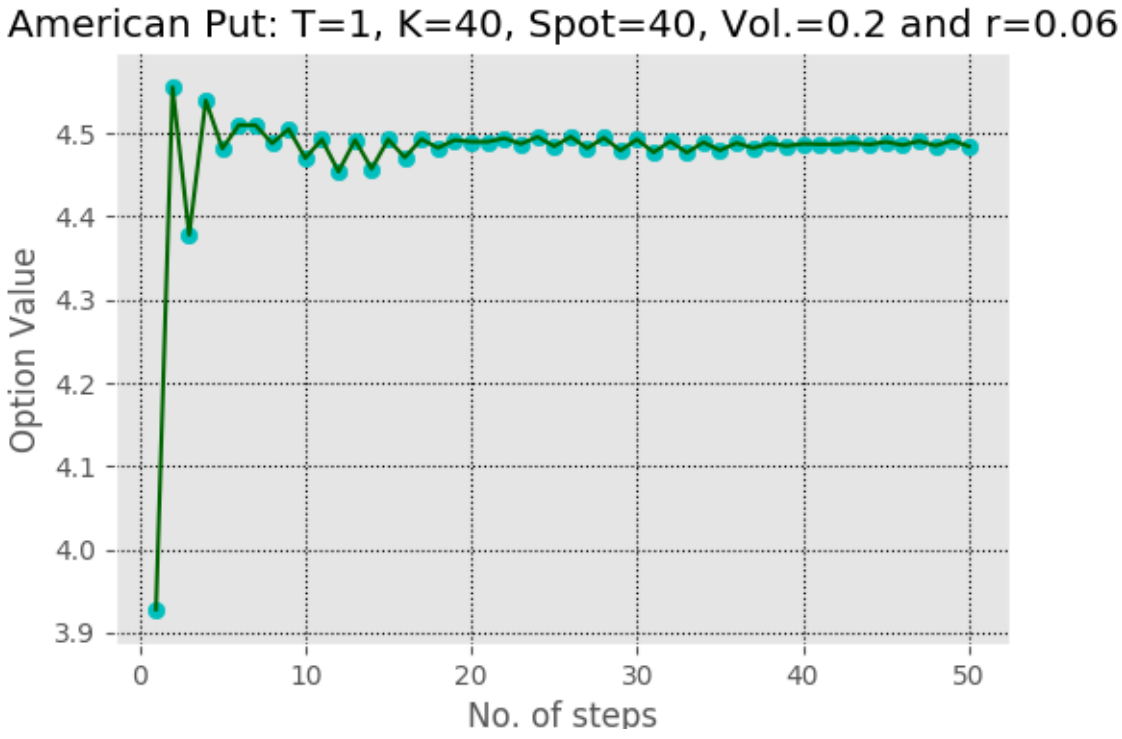

FIGURE 3.3: Price for a american put option based on the binomial model, where the independent variable is the number of time-steps. Vol. is an abbreviation for volatility.

like american options. Even though we assume the stock price moves at discrete jumps instead of the classical Black-Scholes continuous time model, it can actually be shown that the CRR binomial model will converge to the continuous time model. Hence the binomial pricing model will be equivalent with the continuous time analytical pricing model derived by Fischer Black and Myron Scholes in the limit for european options for sufficient small time steps.

Note for computational resources the path-independence payoff for the american put makes the tree recombining is important, so there is only N+1 terminal nodes at maturity. If the derivative was path-dependent e.g. an asian option, then we have a non-recombinin tree and 2^N terminal nodes where needed. This is a computational inefficient, which explains that the binomial model should not be used for path-depending derivatives. The problem with a derivative with several underlying e.g. a basket option is also the increasing number of nodes, because now you have 2^d possible one-step transitions. We will show below that the intuitive CRR binomial model can be extended to higher dimensions, but the model suffers from the curse of dimensionality.

3.2 Lattice Approach For Multivariate Contingent Claims

We follow the approach in (Boyle, Evnine, and Gibbs, 1989) (BEG method), because it is the natural extension of the Cox Ross Rubinstein model (section 3.1) for multivariate contingent claims. The idea as in the one dimensionel case is to approximate the system of underlying processes (assumed to be GBMs) with a discrete multivariate binomial lattice. The advantage is that for exotic options like the rainbow options the valuation of european put options is readily extented to the american put options and has high accuracy. The (Boyle, Evnine, and Gibbs, 1989) has its limitation in terms of number of underlyings and for path dependent options (see section 3.1), but it is very intuitive and extends easily to american options. The problem with increasing the number of underlyings is that the number of one-step transition at each node is 2^d and the total number of terminal nodes after N steps is $(N+1)^d$ for path-independent derivatives, which means for high dimensional problems the computational resources become an issue with this discrete approximation approach. This makes the BEG method undesirable for higher dimensions than three so we will focus on the two dimensional case. Another problem with three or more underlyings are that some one-step transition probabilities can turnout negative, which makes the model nonsense in those cases.

The model we want to approximate is the bivariate lognormal distribution, because we assume the Black Scholes model to describe the evolution of the two risky assets (section 2.2). We restrict ourselfes to the assumptions 2.15 given in the classical (Black and Scholes, 1973), hence for risk neutral pricing the SDE for the risky assets are:

$$dS_i(t) = S_i(t)r(t)dt + S_i(t)\sigma_i(t)dW^{\mathbb{Q}}(t) \quad for\ i = 1, 2, \ldots, d$$

We divide the time from inception to maturtiy (length T) into N equidistant intervals with length Δt, because we want the jump distribution to approximate the continuous time multivariate lognormal distribution. Each time interval has a jump size defined in terms of the volatility and the length of the interval:

$$u_i = \exp(\sigma_i\sqrt{\Delta t}) \quad and \quad u_i \cdot d_i = 1 \quad for\ i = 1, 2, \ldots, d$$

The u_i and d_i are the multiplication factor for the i'th stock, where the former is a jump up and the latter is a jump down for the stock. What is the probability that the stock jumps up or down? The probabilities are chosen such that the characteristics functions are equal for small time steps Δt (see p. 245-246 in (Boyle, Evnine, and Gibbs, 1989) for details). The probabilities for the model with two underlying risky assets:

$$\begin{aligned}
p_1 = p_{uu} &= \frac{1}{4}\left(1 + \rho + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2}\right)\right) \\
p_2 = p_{ud} &= \frac{1}{4}\left(1 - \rho + \sqrt{\Delta t}\left(\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2}\right)\right) \\
p_3 = p_{du} &= \frac{1}{4}\left(1 - \rho + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} + \frac{\mu_2}{\sigma_2}\right)\right) \\
p_4 = p_{dd} &= \frac{1}{4}\left(1 + \rho + \sqrt{\Delta t}\left(-\frac{\mu_1}{\sigma_1} - \frac{\mu_2}{\sigma_2}\right)\right)
\end{aligned} \tag{3.5}$$

The correlation ρ between the two assets are assumed to be constant and $\mu_i = r - \frac{1}{2}\sigma_i^2$. We have illustrated below a two-dimensional lattice, where we see that the number of nodes at maturity is $(1+N)^d$.

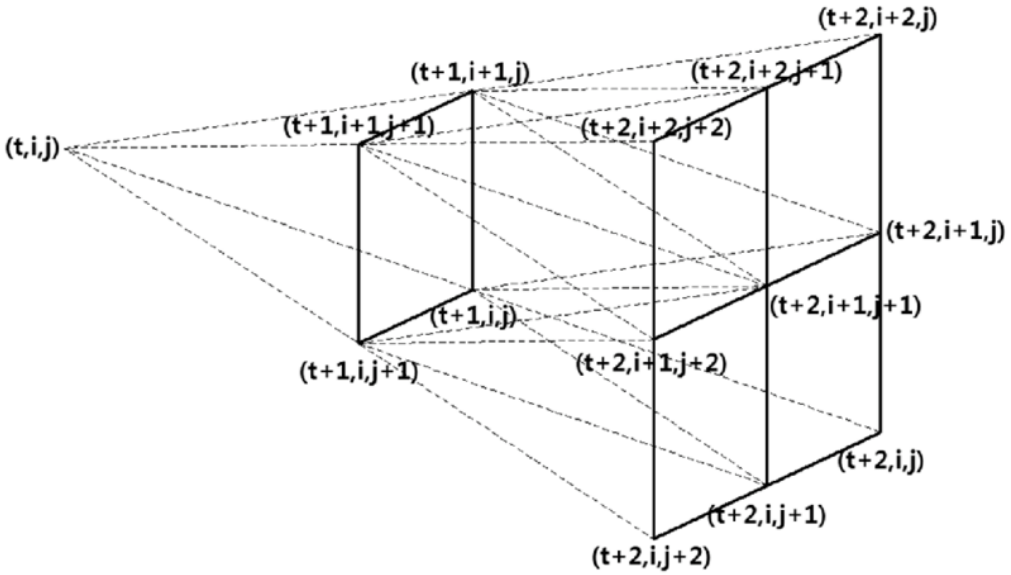

FIGURE 3.4: Evolution of binomial model with two underlying risky asset, where t is time, i is number of up movement for S_1 and j is number of up movement for S_2 (**Kim02**).

After the construction of the evolution of the underlyings, we can like in the one dimensionel CRR model recursively working backward in the multidimensional binomial lattice. For the european option the recursive formula is:

$$V_{i_1,\ldots,i_d}(t) = \exp(-r\Delta t)\Big(p_1 V_{i_1+1,\ldots,i_d+1}(t+1) + \cdots + p_{2^d} V_{i_1,\ldots,i_d}(t+1)\Big)$$

For the american option we approximate it with the Burmudan option with N-1 decision points, hence we have the possibility of exercising between inception and maturity of the contract at the decision points hence:

$$V_{i_1,\ldots,i_d}(t) = \max\{\Phi(t,\boldsymbol{S}(t)), \exp(-r\Delta t)\Big(p_1 V_{i_1+1,\ldots,i_d+1}(t+1) + \cdots + p_{2^d} V_{i_1,\ldots,i_d}(t+1)\Big)\}$$

With the recursive formulas we can valuate multivaraite contigent claims for a variety of exotics including the american put option with several underlying risky assets. The increasing dimensions also increase the number of one-step transition probabilities, hence the (Ekvall, 1996) approach (NEK) tries with setting all the probabilities equal to 0.5 and than determine the jump sizes. The NEK approach overcome the issue with negative probabilities for higher dimensions and following (Ekvall, 1996) the algorithm seems also to have faster convergence. The NEK and BEG approaches are both good for low dimensionel option problems, but both still suffers from the curse of dimensionality. The simulations methods on the otherhand has somewhat an advantage for higher dimesions, but e.g. the LSM regression does not scale well for high dimensional rainbow options. We choose to present the BEG approach,

because it is a natural extension of the already presented CRR model, which we already build inituition opon. The key for the BEG approach is that we approximate a multivariate lognormal distribution with a discrete distribution.

3.3 Least Square Monte Carlo Method

From above the binomial model is not suitable for path-dependent contingent claims like an asian option and high dimensional multivariate contingent claims. Simulation methods overcome somewhat this issue. The first monte carlo methods was to use pure simulation techniques, these methods overcome the issue of path-dependent option, but they still suffered the curse of dimensionality. To solve the dimensionality problem the LSM was born, where the idea is combine simulation with regression. The methods presented in this

A pure simulation technique have three ingredients a simulation based on the assumption of the underlying asset(s) distribution to price potential future prices. From the simulation of the underlying(s) use the contract function to get the cash flow, then discount back to present value and then average over the simulated paths. The approach is suitable for the asian option, because by simulatation you have the whole path and averaging is straightforward. On the other hand the binomial model is computational fast and accurate for american options, because by discretization the underlying stock the algorithm give an effective way to get intrinsic and expected continuation values. The pure simulation method would not be ideal for american options, because a each decision point for the american option the pure simulation would need to simulate a new set of paths to estimated the expected continuation value. For reference of the pure simulation method is chapter 10 in (OVERHAUS et al., 2007).

The computer is discrete by nature, so we approximate the american put option with a bermudan put option. The N time points or exercise points chosen between inception and maturity are equidistant time steps and by chosen N sufficient large the bermuda option approximate the american option. Hence the simulatation methods use the setup in section 2.4.3, because we discretize the decision points. We use again dynamic programming to calculate the price of the american put option. The problem with the pure simulation approach is the computational burden to evaluate the expected continuation value at each exercise data. The Longstaff and Schwartz LSM algorithm overcomes the exponential growing computation burden in pure Monte Carlo simulation by using regression to calculate the expected continuation value.

3.3.1 The Algorithm

Before solving the optimal stopping problem numerically, we set up the mathemeatical framework for solving the problem inspired by (Clément, Lamberton, and Protter, 2001). The optimal problem in discrete time is:

$$\sup_{\tau \in \mathcal{T}(0,\dots,T)} E^{Q}[G(S(\tau))] \tag{3.6}$$

where $\mathcal{T}(0,\dots,T)$ is a class of all $(0,\dots,T)$-valued stopping times and $\Big(S(0), S(t_1), \dots, S(t_N) = S(T)\Big)$ is the underlying stochastic markov process describing e.g. stock price, volatility, average stock price, etc. The markov chain $(S_{t_n})_{n=0,\dots,N}$ has state space $(E, \mathcal{E})$.

Remember the discrete optimal value process is given by

$$U(t_n) = \operatorname*{ess\,sup}_{\tau \in \mathcal{T}(t_n,\ldots,t_N)} E^Q[G(S(\tau))|\mathcal{F}_n]$$

The markov property implies that

$$E^Q[G(S(\tau_{t_{n+1}}))|\mathcal{F}_{t_n}] = E^Q[G(S(\tau_{t_{n+1}}))|S(t_n)] = f(S(t_n))$$

and we assume the initial state to be $S(0) = s$ and deterministic. We solve equation (3.6) by the theory presented in section 2.4.3, hence the dynamic programming principle on the optimal policy is

$$\begin{cases} \tau_{t_N} = t_N \\ \tau_{t_n} = t_n \cdot 1_{\{G(S(t_n)) \geq E^Q[G(S(\tau_{t_{n+1}}))|\mathcal{F}_{t_n}])\}} + \tau_{t_{n+1}} \cdot 1_{\{G(S(t_n)) < E^Q[G(S(\tau_{t_{n+1}}))|\mathcal{F}_{t_n}])\}} \end{cases} \quad for\ n = 0, \ldots, N-1 \tag{3.7}$$

Equation (3.7) involves many conditions expectations, hence we need an effective algorithm for evaluating them. The first approaches for solving the optimal stopping problem with pure monte carlo was computational expensive, because with pure simulation methods the paths required for evaluate the conditional expectations increases exponential with decision points (chapter 10 (OVERHAUS et al., 2007)). The solution to reduce the computational burden is to use least square regression to calculate the expected continuation value suggested e.g. in (Longstaff and Schwartz, 2001; Tsitsiklis and Roy, 1999).

The LSM algorithm approximate the condition expectation with respect to $S(t_n)$ by orthogonal projection on the state-space genereated by finite number of functions of $S(t_n)$. Define a sequence $(e_j(s))_{j\geq 1}$ of real measurable functions defined on E and satisfying:

- The sequence $(e_j(S(t_n)))_{j\geq 1}$ is total in $L^2(\sigma(S(t_n)))$ for $n = 1, \ldots, N-1$.
- if $\sum_{j=1}^m \lambda_j e_j(S(t_n)) = 0$ *a.s.* then $\lambda_j = 0$ for $n = 1, \ldots, N-1$, $m \geq 1$ and $j = 1, \ldots, m$

By defining the vector space generated by $(e_j(s))_{j=1,\ldots,m}$ we denote the orthogonal projection from $L^2(\Omega)$ onto the vector space by $P^m_{t_n}$. We approximate the expected continuation value by the projection

$$\begin{cases} \tau^{[m]}_{t_N} = t_N \\ \tau^{[m]}_{t_n} = t_n \cdot 1_{\{G(S(t_n)) \geq P^m_{t_n}(G(S(\tau_{t_{n+1}})))\}} + \tau_{t_{n+1}} \cdot 1_{\{G(S(t_n)) < P^m_{t_n}(G(S(\tau_{t_{n+1}})))\}} \end{cases} \quad for\ n = 0, \ldots, N-1 \tag{3.8}$$

Where

$$P^m_{t_n}(G(S(\tau^{[m]}_{t_{n+1}}))) = \alpha^m(t_{n+1}) \cdot e^m(S(t_n)) \quad for\ n = 1, \ldots, N-1$$

and the $\cdot$ on the right hand side is the usual inner product in $\mathbb{R}^m$ and $e^m = (e_1, \ldots, e_m)$. By solving the above dynamic programming equation the time 0 approximate price of the Bermudan option is:

$$U^m(0) = \max\{G(S(0)), E^Q[G(S(\tau^{[m]}_{t_1}))]\}$$

Where $G(S(0))$ is deterministic by assumption, but $E^Q[G(S(\tau_{t_1}^{[m]}))]$ needs to be evaluated numerically, where in the LSM algorithm the methods for evaluation the projection is monte carlo simulation.

From the assumption about the distribution of the underlying state space we simulate K independent paths $S^{(1)}(t_n), S^{(2)}(t_n), \ldots, S^{(k)}(t_n), \ldots, S^{(K)}(t_n)$ of the Markov chain $S(t_n)$ with gain process $G(S^{(k)}(t_n))$ for $k = 1, \ldots, K$ and $n = 0, \ldots, N$. The least square estimator $\alpha^{(m,K)}(t_n) \in \mathbb{R}^m$ for the simulated paths of the stochastic process S.

$$\alpha^{(m,K)}(t_n) = \underset{a \in \mathbb{R}^m}{\operatorname{argmin}} \frac{1}{K} \sum_{k=1}^{K} \left(G(S^{(k)}(\tau_{t_{n+1}}^{k,m,K})) - a \cdot e^m(S^{(k)}(t_n)) \right)^2 \quad for\ n = 1, \ldots, N-1 \tag{3.9}$$

Where the recursively estimated stopping time is defined by:

$$\begin{cases} \hat{\tau}_{t_N}^{k,m,K} = t_N \\ \hat{\tau}_{t_n}^{k,m,K} = t_n \cdot 1_{\{G(S^{(k)}(t_n)) \geq \alpha^{m,K}(t_n) \cdot e^m(S^{(k)}(t_n))\}} + \hat{\tau}_{t_{n+1}}^{k,m,K} \cdot 1_{\{G(S^{(k)}(t_n)) < \alpha^{m,K}(t_n) \cdot e^m(S^k(t_n))\}} \end{cases} \quad for\ n = 0, \ldots, N-1 \tag{3.10}$$

An example of polynomial linear on each decision points is illustrated in figure 3.5, where the blue line is the estimated expected continuation value. From following the optimal stopping stragegy by dynamic programming we derive the approximation for $U^m(0)$ from the simulated paths. The time 0 approximate price of the Bermudan option is

$$U^{m,K}(0) = \max\{G(S(0)), \frac{1}{K} \sum_{k=1}^{K} G(S^{(k)}(\hat{\tau}_{t_1}^{k,m,K}))]\} \tag{3.11}$$

3.3.2 American Put

The optimal stopping problem for an american put

$$P(0) = \sup_{\tau \in \mathcal{T}(0,\ldots,T)} E^Q[e^{-r\tau} \cdot \max\{K - S(\tau), 0\}] \tag{3.12}$$

can be solved with the LSM algortihm. The stock values are modelled via black scholes theory (see path for GBM figure 2.2), hence the simulated evolution for the stock under the risk neutral valuation is given by:

$$S_i(t) = S_i(0) \cdot \exp\left(\sum_{j=1}^{d} (\sigma_{i,j} W_j(t) - \frac{1}{2}\sigma_{i,j}^2 t) + rt \right) \quad for\ i = 1, \ldots, d$$

The stock paths are simulated from inception up to maturty with N-1 decision dates. The focus in this section is on a univariate contingent claim and for convenience we assume the risk free interest rate and volatility is constant. Like in the binomial model, we work backward from maturity to inception at each exercise dates to decide the optimal stopping time.

The dynamic programming principle on optimal policy gives the first optimal stopping time. In our setting we regress the expected payoff by continuation of the contract and compare it to the intrinsic value. The dependent variable in the regression is the expected value of continuation and the independent variables is a set of

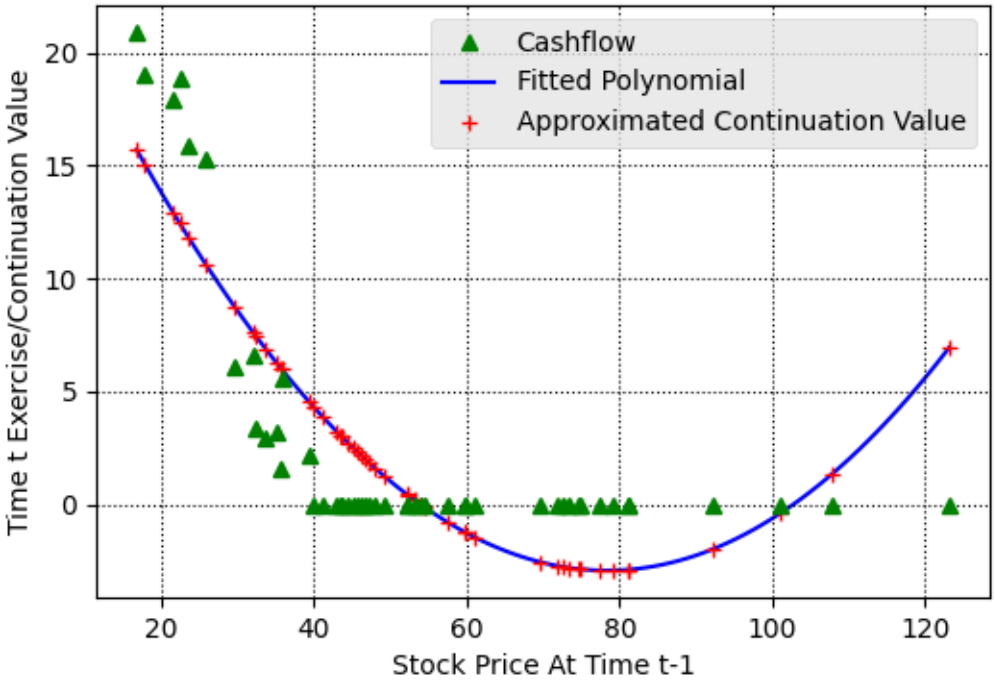

FIGURE 3.5: By zooming in on a specific point of time in backward induction approach, we see how the algorithm regress the continuation value

orthogonal basis functions in $L^2(\sigma(S(t_n)))$ of the simulated paths. Typical choices for basis functions could be weighted Laguerre -, Hermit -, and Jacob polynomials. The weighted Laguerre polynomial is given by

$$\begin{aligned} e_0(S) &= \exp(-S/2) \\ e_1(S) &= \exp(-S/2)(1-S) \\ e_2(S) &= \exp(-S/2)(1-2S+S^2/2) \\ &\vdots \\ e_j(S) &= \exp(-S/2)\frac{e^S}{j!}\frac{d^j}{dS^j}(S^j e^{-S}) \end{aligned}$$

This kind of regression is a nonlinear expansion of the linear model. We define regressed conditional expectation by:

$$\Psi(S;\alpha) = \sum_{j=0}^{p} \alpha_j \cdot e_j(S)$$

where α is the coefficients for the regression, e is the basis function, where the argument is the underlying markovian process S. By using this iterative method, we arrive at the pathwise optimal stopping policy, where in figure 3.6 the optimal stopping times are shown. The figure illustrates that the option only can be exercised once, hence the gray lines after the triangles.

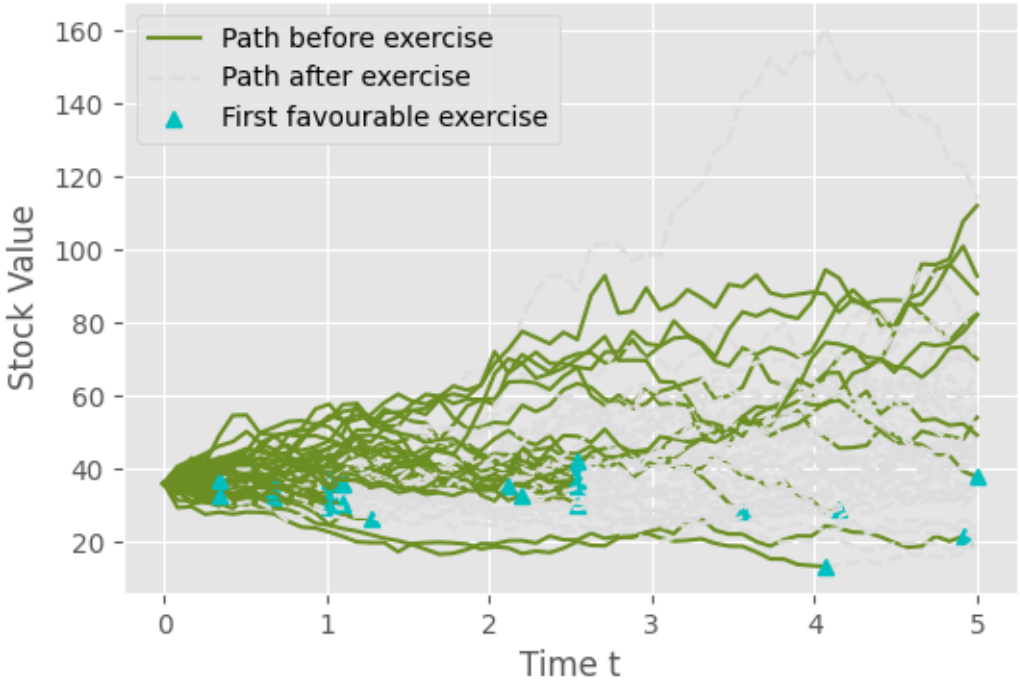

FIGURE 3.6: The optimal stopping decisions by the Least Square Monte Carlo Method

3.3.3 Convergence

In the rigorous approach in (Clément, Lamberton, and Protter, 2001) they show convergence results for the optimal value process or the snell envelope U. We will present that $U(0)^{m,K}$ converges almost surely to $U(0)^m$ for K goes to infinity, i.e. the approximate value process by simulation and regression on a finite set of functions converge to the approximated value process with truncated orthogonal basis by letting the sample size go to infinity. Furthermore it can be shown that $U(0)^m$ converge to $U(0)$ for m goes to infinity. The second result is that the regressed value function converge to the expected continuation value by letting the number of basis function goes to infity. The latter result is shown using the expected continuation values.

Theorem 3.3. *Assume the sequence* $(e_j(S(t_n)))_{j\geq 1}$ *is total in* $L^2(\sigma(S(t_n)))$ *for* $n = 1, \ldots, N-1$. *Then for* $n = 0, \ldots, N$ *we have*

$$\lim_{m\to+\infty} E^Q[G(S(\tau_{t_n}^{[m]}))|\mathcal{F}_{t_n}] = E^Q[G(S(\tau_{t_n}))|\mathcal{F}_{t_n}]$$

in L^2

Proof. The proof is given by induction, where the orthogonal basis is total in L^2 is important, because $||P_{t_n}^m(E^Q[G(S(\tau_{t_{n+1}}))|\mathcal{F}_{t_n}]) - E^Q[G(S(\tau_{t_{n+1}}))|\mathcal{F}_{t_n}]||_2 \to 0 \quad for\ m \to \infty$. (more details on p. 6-7 (Clément, Lamberton, and Protter, 2001)) ■

The former result is also shown in (Clément, Lamberton, and Protter, 2001).

Theorem 3.4. *Assume the sequence* $(e_j(S(t_n)))_{j\geq 1}$ *is total in* $L^2(\sigma(S(t_n)))$ *for* $n = 1, \ldots, N-1$ *and if* $\sum_{j=1}^{m} \lambda_j e_j(S(t_n)) = 0$ *a.s. then* $\lambda_j = 0$ *for* $n = 1, \ldots, N-1$, $m \geq 1$ *and* $j = 1, \ldots, m$. *Futhermore assume that* $Q(\alpha_{t_n} \cdot e(S(t_n)) = G(S(t_n))) = 0$.
Then $U^{m,K}(0)$ *converges almost surely to* $U^m(0)$ *as K goes to infinity.*
The proof is out of scope for this thesis, see the article (Clément, Lamberton, and Protter, 2001) for a proof in details.

The two convergence results shows the convergence for the LSM algorithm, hence the LSM will approximate the optimal value process well for sufficient large sample sets and enough basis functions.

3.3.4 Upper Bound

By the results from (Andersen and Broadie, 2004) they provide a algorithm for computing a lower and upper bound for the american option price, where the lower bound is generated by approximating the continuation value. They show with martingale methods that the interval becomes smaller when the lower bound approaches the true value. The true value will always by higher than the approxmixation, because the true value is when you exercise optimal at each decision point. Hence to compare approximation method on the continuation value a higher lower bound indicate a better approximation to the true value.

Given the optimal exercise boundary is only an estimate, both the methods underestimate the "true value" of the option.

A simple comparison would be whichever method produced higher price for the option is better.

For this comparison to make sense, you could

re-use underlying stock simulation across both the methods. make sure variance of price produced is reasonably low for both.

The "better" value of the two is still a lower bound and doesn't really throw information on how big the error is.

You could implement dual method to produce upper bound and thus a range for the true option price. The regression performed in LSM is only for in money path for improving the algorithm, wher

The LSM approach gives a lower bound for the true price of the option given optimal stopping choice:

Proposition 3.5. ***Lower Bound To True Value:*** *For any finite choice of M, K, and vector* $\theta \in \mathbb{R}^{M\times(K-1)}$ *representing the coefficients for the M basis functions at rach of the K-1 early exercise dates, let* $LSM(\omega; M, K)$ *denote the discounted cash flow resulting from the following the LSM rule of exercising when the immediate exercise value is positive and greater tahn or equal to* $\hat{F}_M(\omega_1; t_k)$ *as defined by* θ. *Then the following inequality holds almost surely,*

$$V(X) \geq \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} LSM(\omega_i; M, K)$$

(p. 124 (Longstaff and Schwartz, 2001))

3.3.5 LSM Extension To Multivariate Contingent Claims

For pricing of bivariate and multivariate contingent claims, we have to account for correlation between assets. The correlation matrix Σ is given by

$$\Sigma = \begin{pmatrix} \rho_{1,1} & \rho_{1,2} & \cdots & \rho_{1,d} \\ \rho_{2,1} & \rho_{2,2} & \cdots & \rho_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{d,1} & \rho_{d,2} & \cdots & \rho_{d,d} \end{pmatrix} = \Sigma = \begin{pmatrix} 1 & \rho_{1,2} & \cdots & \rho_{1,d} \\ \rho_{2,1} & 1 & \cdots & \rho_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{d,1} & \rho_{d,2} & \cdots & 1 \end{pmatrix} \tag{3.13}$$

The correlation between assets are given by $(\rho_{i,j})_{i \neq j}$ and each asset has correlation 1 with itself. We assume that $(\rho)_{i \neq j} \in (-\frac{1}{d-1}, 1]$, because then the correlation matrix is a real, symmetric and positiv definit, hence Cholesky factorization can be utilized $\Sigma = LL^T$ where L is a lower triangular matrix. From the decomposition it is easy to simulate correlated assets in d-dimensional Black-Scholes model:

$$dS_i(t) = S_i(t)rdt + S_i(t)\sigma_i L_{i,.} d\boldsymbol{W}(t) \quad i \in \{1, 2, \ldots, d\} \tag{3.14}$$

Where $\boldsymbol{W}$ has values in $\mathbb{R}^d$ and $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_d)$ is vector of volatilities.

By simulating the paths according to the black scholes dynamic, it is straightforward to apply the LSM, because it follows the same algortihm just with a different contract function.'

3.4 Closed Form Solutions For European Exotic Options

Most exotic options require numerical methods, but in some special cases there exist a closed form solution. We will look at some of them in this section, where the purpose is to provide benchmarks for the numerical methods. Furthermore we explore the boundaries of closed form solutions and show applications of martingale theory. Thorughout the financial model and assumptions given in section 2.2 will be assumed. We derive closed form solutions to european call and put options depending on several variables, for simplicity we will focus on pricing options with 2 or 3 underlying stocks. We apply the intuition given in (Johnson, 1987) and the results given in (Ouwehand, 2006). The exotic contingent claims we will consider are the geometric mean -, maximum - and minimum call option.

3.4.1 Geometric Basket Call Option

For a geometric basket call option the contract function is given by:

$$\Phi(S(T)) = \max\{(\prod_{i=1}^{n} S_i(T))^{\frac{1}{n}} - K, 0\}$$

The key to derive a closed form solution is the known result that the sum of normal random variables are multivariate normal distributed. This implies that the product of lognormal random variables are multivariate log-normal distributed. Since:

$$\exp(x+y) = \exp(x) \cdot \exp(y)$$
$$\text{and}$$
$$X \sim \mathcal{N}(\mu, \sigma^2) \Rightarrow Y = \exp(X) \sim LN(\mu, \sigma^2)$$

Remember the assumption in section 2.2 that the stocks price process follows a GBM, hence:

$$(\prod_{i=1}^{d} S_i(T))^{\frac{1}{d}} = (\prod_{i=1}^{d} S_i(0))^{\frac{1}{d}} \exp\left((r - \frac{1}{2d}\sum_{i=1}^{d}\sigma_i^2)T + \frac{1}{d}\sum_{i=1}^{d}\sigma_i W_i(T)\right) \tag{3.15}$$

By defining

$$\tilde{\sigma} = \frac{1}{d}\sqrt{\sum_{i=1}^{d}\sigma_i^2 + 2\sum_{i\neq j}\rho_{i,j}\sigma_i\sigma_j} \tag{3.16}$$

$$F = (\prod_{i=1}^{d} S_i(0))^{\frac{1}{d}} \exp\left((r - \frac{1}{2d}\sum_{i=1}^{d}\sigma_i^2)T + \frac{1}{2}\tilde{\sigma}^2 \cdot T\right) \tag{3.17}$$

$$\epsilon = \frac{\frac{1}{d}\sum_{i=1}^{d}\sigma_i W_i(T)}{\tilde{\sigma}\sqrt{T}} \sim \mathcal{N}(0,1) \tag{3.18}$$

Rewrite equation (3.15) by above definitions

$$(\prod_{i=1}^{d} S_i(T))^{\frac{1}{d}} = F \cdot \exp\left(-\frac{1}{2}\tilde{\sigma}^2 \cdot T + \tilde{\sigma}\sqrt{T}\epsilon\right)$$

This expression is one dimensional and is the stardard GBM solution with zero drift and spot F. This has a known solution with Black-Scholes theory (section 2.3) and

the geometric mean call option has price

$$\Pi(t,\mathcal{X}) = \exp(-r\cdot(T-t))\Big(FN(d_1) - KN(d_2)\Big)$$

where $d_1 = \frac{\ln(\frac{F}{K})+\frac{1}{2}\tilde{\sigma}^2 T}{\tilde{\sigma}\cdot\sqrt{T}}$ and $d_2 = d_1 - \tilde{\sigma}\sqrt{T}$

The fact that the sums of normals is multivariate normal distributed makes the geometric mean option easy to price in the Black-Scholes model, because the high dimensional problems can be treated as 1-dimensional problem like above with the european geometric mean call option.

3.4.2 Options On The Maximum Or The Minimum Of Several Assets

Here we restrict ourselves to consider the case with three underlying stocks like in (Boyle, Evnine, and Gibbs, 1989; Ouwehand, 2006), but the formula can be generalized to higher dimensions. The contract functions we consider are:

- Best of assets or cash: $\Phi(S(T)) = \max\{S_1, S_2, \ldots, S_d, K\}$
- Call on max: $\Phi(S(T)) = \max\{\max(S_1, S_2, \ldots, S_d) - K, 0\}$

Assume WLOG d=4 to avoid cumbersome calculations and notation. The section will heavily use the martingale framework developed in section 2.2 and stochastic calculus (Appendix B) to value these exotic options. The key is to choose the numeraire to a risky assets instead of the bank account. By results from section 2.2 the processes are still Q-martingales given the numeraire is strictly postive. Under the asssumption of a arbitrage free and complete market it follows from risk neutral valuation formula(RNVF).

$$S_0(t)E_t^{Q_0}[\frac{X}{S_0(T)}] = S_1(t)E_t^{Q_1}[\frac{X}{S_1(T)}]$$

This show that changing the numaraire does not change the price of the above derivative.

3.4.2.1 Best Of Assets Or Cash

The best of assets derivative will provide the foundation for pricing call on maximum or minimum of several assets. The best of assets derivative pay out the price of the most valuable asset. The idea is then to set up 4 cases, where in each case a new asset gives a payout if it is the most valuable out of the four cases. This can be written with a indicator, where the payoff for the i'th asset is:

$$S_i(T)\cdot 1_{\{S_i(T)>S_j(T):i\neq j\}} \quad i = \{1,2,3,4\} \tag{3.19}$$

From equation (3.19) we see that the i'th assets is only different from zero, when it is the most valuable asset. The best of asset derivative is then a sum of 4 terms (remember d=4), where each term is equation (3.19) for a specific asset i. Hence the best of asset derivative can be consider as a sum of 4 derivatives, where we apply RNVF (proposition 2.13) for each of them.

Let us first consider i=1 and we set S_1 to be the numeraire asset with martingale measure $\mathbb{Q}_1$. Then by RNVF:

$$\begin{aligned}\Pi_1(t,\mathcal{X}) &= S_1(t)E_t^{Q_1}[1_{\{S_1(T)>S_2(T),S_1(T)>S_3(T),S_1(T)>S_4(T)\}}] \\ &= S_1(t)Q_1[\ln(\frac{S_2(T)}{S_1(T)})<0,\ln(\frac{S_3(T)}{S_1(T)})<0,\ln(\frac{S_4(T)}{S_1(T)})<0]\end{aligned} \tag{3.20}$$

From above discussion the derivative price can be found by cycling through the 4 derivatives and add together the prices of the four derivatives. The fair price for best of assets option $\Pi_{max}(t,\mathcal{X})$ is then given as the sum of the 4 derivatives. The above derivative equation (3.20) needs to be evaluate, therefore we seek to evaluate the probaility under the Q-martingale measure. By Ito's lemma (see A.1) the discounted process with stock j is

$$d(\frac{S_i(t)}{S_j(t)}) = \frac{S_i(T)}{S_j(T)}\cdot(a_i-a_j)dW^{Q_j}(t) \tag{3.21}$$

where $dW^{Q_j}(t)$ is standard d-dimensional Q_j-Wiener process and a_i is the i'th row of the covariance matrix. From the known solution of the GBM we have the distribution.

$$\ln(\frac{S_i(T)}{S_j(T)}) \sim \mathcal{N}\left(\ln(\frac{S_i(t)}{S_j(t)}) - \frac{1}{2}\sigma_{i/j}^2\cdot(T-t), \sigma_{i/j}^2(T-t)\right)$$

where $\sigma_{i/j} = (a_i - a_j)$ and $\sigma_{i/j}^2 = \sigma_i^2 + \sigma_j^2 - 2\rho_{ij}\sigma_i\sigma_j$.

Besides using the definition for d_1 and d_2 in proposition 2.16 we define:

$$d_1^{i/j} = \frac{1}{\sigma_{i/j}\cdot\sqrt{T-t}}\cdot\left(\ln(\frac{S_i(t)}{S_j(t)}) + \frac{1}{2}\sigma_{i/j}^2\cdot(T-t)\right) \tag{3.22}$$

$$d_2^{i/j} = d_1^{i/j} - \sigma_{i/j}\sqrt{T-t} \tag{3.23}$$

The correlation between $\frac{S_i(T)}{S_k(T)}$ and $\frac{S_j(T)}{S_k(T)}$ is:

$$\rho_{ij,k} := \frac{(a_i-a_k)\cdot(a_j-a_k)}{||a_i-a_k||\cdot||a_j-a_k||} \tag{3.24}$$

$$= \frac{\rho_{ij}\sigma_i\sigma_j - \rho_{ik}\sigma_i\sigma_k - \rho_{kj}\sigma_k\sigma_j + \sigma_k^2}{\sqrt{(\sigma_i^2+\sigma_k^2-2\rho_{ik}\sigma_i\sigma_k)\cdot(\sigma_j^2+\sigma_k^2-2\rho_{jk}\sigma_j\sigma_k)}} \tag{3.25}$$

Hence:

$$Q_1[\ln(\frac{S_2(T)}{S_1(T)})<0,\ln(\frac{S_3(T)}{S_1(T)})<0,\ln(\frac{S_4(T)}{S_1(T)})<0] = N_3(-d_2^{2/1},-d_2^{3/1},-d_2^{4/1},\rho_{23,1},\rho_{24,1},\rho_{34,1})$$

Cycling through each derivative, we get:

$$\begin{aligned}\Pi_{max}(t,\mathcal{X}) = {} & S_1(t)N_3(-d_2^{2/1},-d_2^{3/1},-d_2^{4/1},\rho_{23,1},\rho_{24,1},\rho_{34,1}) \\ & + S_2(t)N_3(-d_2^{1/2},-d_2^{3/2},-d_2^{4/2},\rho_{13,2},\rho_{14,2},\rho_{34,2}) \\ & + S_3(t)N_3(-d_2^{1/3},-d_2^{2/3},-d_2^{4/3},\rho_{12,3},\rho_{14,3},\rho_{24,3}) \\ & + S_4(t)N_3(-d_2^{1/4},-d_2^{2/4},-d_2^{3/4},\rho_{12,4},\rho_{13,4},\rho_{23,4})\end{aligned} \tag{3.26}$$

We can extend the above result to best of assets and cash by letting $S_4(t) = K\exp(-r(T-t))$, where K do not have any volatilty and also independent of the other assets, hence (3.26) becomes:

$$\begin{aligned}\Pi_{max}(t,\mathcal{X}) &= S_1(t)N_3(-d_2^{2/1},-d_2^{3/1},d_1^1,\rho_{23,1},\rho_{24,1},\rho_{34,1})\\ &+ S_2(t)N_3(-d_2^{1/2},-d_2^{3/2},d_1^2,\rho_{13,2},\rho_{14,2},\rho_{34,2})\\ &+ S_3(t)N_3(-d_2^{1/3},-d_2^{2/3},d_1^3,\rho_{12,3},\rho_{14,3},\rho_{24,3})\\ &+ K\cdot\exp(-r(T-t))N_3(-d_2^1,-d_2^2,-d_2^3,\rho_{12},\rho_{13},\rho_{23})\end{aligned} \tag{3.27}$$

3.4.2.2 Call On Max And Call On Min

From the price of the best of asset or cash option equation (3.27) the call max option price can be derived. Note that

$$\max\{\max\{S_1,S_2,S_3\}-K,0\} = \max\{\max\{S_1,S_2,S_3\},K\}-K = \max\{S_1,S_2,S_3,K\}-K$$

The call on max option is then a best of asset or cash option subtracted the strike. Realizing that fact it is easy to price the call on max:

$$\begin{aligned}\Pi_{cmax}(t,\mathcal{X}) &= S_1(t)N_3(-d_2^{2/1},-d_2^{3/1},d_1^1,\rho_{23,1},\rho_{24,1},\rho_{34,1})\\ &+ S_2(t)N_3(-d_2^{1/2},-d_2^{3/2},d_1^2,\rho_{13,2},\rho_{14,2},\rho_{34,2})\\ &+ S_3(t)N_3(-d_2^{1/3},-d_2^{2/3},d_1^3,\rho_{12,3},\rho_{14,3},\rho_{24,3})\\ &- K\exp(-r(T-t))\cdot\left(1-N_3(-d_2^1,-d_2^2,-d_2^3,\rho_{12},\rho_{13},\rho_{23})\right)\end{aligned} \tag{3.28}$$

To derive put max we can utilize a put-call-parity (see page 6 (Ouwehand, 2006)), but it takes a different form than the one presented in 2 (see 2.17). The relationship for the exotic call options:

$$V_c(K) + K\exp(-r\cdot(T-t)) = V_p(K) + V_c(0) \tag{3.29}$$

Where $V_c(K)$ is the value of the exotic call option and $V_c(0)$ is the best of assets call option.

The exotic european options serve as benchmark for the multivariate lattice approach, because if the lattice appproach is close to the european benchmark then it is reasonable to expect a close estimat to the american option. The above section is also a presentation of the martingale approach versatility.

Chapter 4

Deep Learning

The chapter is inspired by (Goodfellow, Bengio, and Courville, 2016; MacKay, 2018) which the interested reader can find more information about deep learning. Deep learning experiences a renaissance, because of the technology improvements in hardware and software. The collection of data has also significant improved the field. Deep learning is a specialized field in machine learning, where you focus on a special architecture of models. Like in machine learning the basic components of a deep learning algorithm are a dataset, cost function, optimazation algorithm and a model. E.g. in the lsm method we assumed the linear model, dataset was the simulated paths, the cost function was the mean square error and the optimazation algorithm was a closed form solution of the normal equations. Deep learning is about studying neural networks which allows for greater flexiblity than standard methods like linear regression. A neural network consists of multiple layer, where the depth tells you how many layers the network has (see figure 4.2), hence the name "Deep learning". All the algorithms applied will be within supervised learning, where we try to fit the best relationship between the features and the response variable. Furthermore all our algorithms will be based on the multilayer perceptrons (MLPs) for regression, therefore most of the chapter present the theory for supervised MLPs regression. The advantage of a multilayer model is that for each layer the updated set of covariates can be more finely tuned to better explain the data making the model extremely flexible. The MLPs is also called feedforward neural networks because the information only travels forward in the neural network, through the input node(s) then through the hidden layer(s) and finally through the output node(s). First we present the basics for machine learning and then specialize the theory to deep learning.

4.1 Machine Learning Basics

The task for machine learning is to learn from data with respect to some performance measure. "A computter program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" ((Goodfellow, Bengio, and Courville, 2016) p. 97). Classical tasks T could be classification or regression, where the two methods differ on the output type. The former has discrete outputs, where regression has continuous output. We are interested finding a price, which is naturally represented as a continuous value hence our task will be to regress the price or continuation value like in LSM. Measuring performance depends on the task, but for regression a typical performance measure is mean squared error (MSE):

$$\frac{1}{K}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2$$

There are other methdos for measuring performance e.g. mean absolute error (MAE). Another measure to quantify the fit is coefficient of determination:

$$R^2 = 1 - \frac{\frac{1}{K}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2}{\frac{1}{K}\sum_{k=1}^{K}(y_k - \bar{y})^2} \quad \text{where } \bar{y} \text{ is the sample mean}$$

Coefficient of determination explains how well the model explains the data compared to the empirical mean. Some care should be taken by comparing models with different capacity, because the coefficient of determination will tend to prefer the larger models. The experience comes from data, where the data can be given with or without target values $\boldsymbol{y}$. In machine learning the task is quite different depending on targets are given or not, hence the algorithms are split into two categories; supervised and unsupervised learning. The terminology supervised comes from a teacher gives you the target values $\boldsymbol{y}$ that the algorithms tries to predict from $\mathbf{X}$.3

Machine learning is not about making overly complex models to fit your data perfectly, because then the model will most likely have poor performace on unseen data. This phenomenon is known as overfitting, but the models can also be too simpel known as underfitting. If machine leearning only cared about performance on the given data for training, then machine learning would be essentially optimization. The key difference is that machine learning wishes to obtain statistical generalization to unseen data. The practical way to evaluate the model is to measure generalization error on a test set. In machine learning the data is split into test data and training data, where the test set can only be used for evaluation after training.

The training data is used for training the model, where the training error tells how the fitted parameters fits to the training data. A unacceptabel high training error could indicate the regresson method could be too simple and underfitting is the issue. For training it can sometimes also be useful to have a validation data set to see how your model generalizes, because the test set cannot be used to make model choices, hence it is common to split the training set into a validation and training set. The validation set is allowed to be used in training and it tries to mimic the generalization error. A common technique for measuring validation errors is "cross-validation", where the data set is split into training and validation sets depending on the cross-validation scheme. The aim for the validation data is to approximate the model performance on unseen data, which is essentially what we want to be within an acceptable range.

After training the model we can evaluate the performance on a test set not seen in training. An unacceptable high test error when evaluating the model and very low training error could be an example of overfitting the model. Hence when training the model, we wish best of both worlds an acceptable training error and making the gap between training and test error acceptable as well. In machine learning the overfitting and underfitting concepts are expressed by the relation of bias-variance tradeoff, where there is a tradeoff between training error and generalization error. In case of overfitting the model has been extensively trained to get a low biases, but the tradeoff is that the model does not generalize well represented by a high variance.

A techinique known as regularization is one approach to avoid overfitting. The regularizer is added to the cost function $J(\theta)$, which is essentially the function to minimize in order to train the model. There is a lot different regularization methods,

where in the MLPs section 4.2 we will present some useful regularization for deep learning models. Besides the parameter estimated within the model, there is a need for model designs (hyperparameters).

Hyperparameter is the parameters not estimated within the model, but rather exogenously given values from the model designer. Some common examples in machine learning and deep learning is the learning rate, batch size, weight decay, model capacity, etc. The choice of the hyperparameters is often more important than the optimization algorithm chosen to estimate the parameters within the model, hence the need for suitable choices for hyperparameters is needed for most machine learning models. The calibration of the model to a specific task is called hyperparameter tuning, where it can be done manually or automatically. The manual choice requires expert knowledge to set the hyperparameter optimal for the given task, where the automatic procedures is often computational expensive. Examples of hyperparameters rutines are random search, grid search and bayesian optimization.

The parameters within the model are found by minimizing the cost function $J(\theta)$. In some cases there exist either a closed form solution or the cost function is convex making the optimization problem easier to handle. The complexity of MLPs makes the cost function nonconvex and iterative optimization procedures are needed. The basic iterative procedure is the gradient descent where the concept is to reapetedly making small moves in parameters toward better configuration. The most popular gradient methods in deep learning are Adam, RMSProp, AdaGrad and stochastic gradient descent (SGD). The methods will be discussed in more details in section 4.2.3. To sum up a machine learning model needs a dataset, a model, a cost function and an optimization algorithm. Understanding of basic machine learning will be the foundation for deep learning.

4.2 Multilayer Perceptrons

The goal of the multilayer perceptrons (MLPs) is to approximate a function $f^*(x)$, where the MLPs defines a mapping $x \in \mathbb{R}^R \mapsto f(x;\theta)$ to approximate $f^*(x)$. The task is to find the best θ such that the approximatation $f(x;\theta)$ is close to the targets measured by a defined cost function $J(\theta)$. With the minimized cost function the goal is to archive statistical generalazation i.e. useful results for test data not used for training.

The first step for MLPs is building the network, where we start with zooming in on a single neuron, which is one node of a directed acyclic graph (see figure 4.2). Note the MLPs is called a feedforward network, because all the connections between the neurons are directed such that the network forms a directed acyclic graph.

4.2.1 A Single Neuron

The single neuron has a number R features of inputs x_r and one output $\hat{y}$ (see figure 4.1).

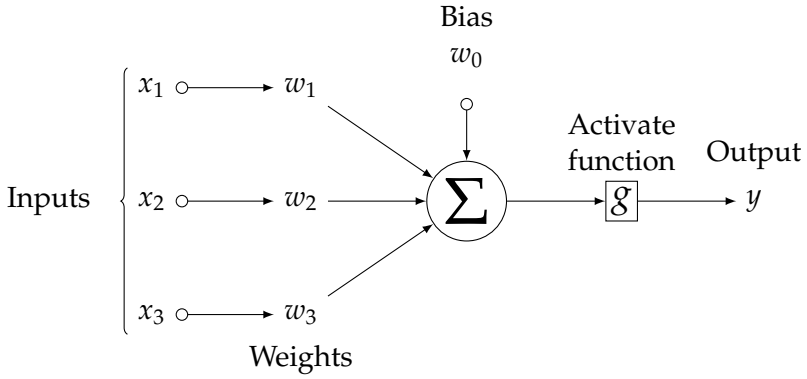

FIGURE 4.1: A single neuron

The function from inputs to output for a single neuron is:

$$\hat{y} = g(w_0 + \boldsymbol{x}^T \boldsymbol{w}) \quad where \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_R \end{pmatrix} \quad and \quad \boldsymbol{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_R \end{pmatrix} \tag{4.1}$$

The $\boldsymbol{w}$ is the weight matrix (this case a vector) and w_0 is the bias term. The term inside the function g is the activation of the neuron and it is a affine transformation denoted:

$$a = w_0 + \boldsymbol{x}^T \boldsymbol{w}$$

The function g is the activation function and it is essential for the flexibility of the MLPs. There exists numerous of activation functions only the imagination is the limit. We will list the most common and discuss them.

4.2.1.1 Activation functions

Activation functions are essential for neural network, because they allow for non-linearities and flexibility. Activation functions apply a non-linear transformation and decide whether a neuron should be activated or not. Without activation functions or the activation function being the identity function $g(a) = a$ the whole network would essentially be a linear regression model. Some popular activation functions are:

- Sigmoid function: $g(a) = \frac{1}{1+\exp(-a)}$

  This is the traditional choice, also called the logistic function. Popular in classification but can suffer from vanishing gradient in deep learning.

- Hyperbolic tangent function: $g(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$

  A scaled and shifted sigmoid function and likewise suffers from the vanishing gradient problem. The range is $(-1, 1)$ and centered at zero. Often used for hidden layers.

- ReLU function: $g(a) = \max(0, a)$.

Rectified Linear Unit is one of the most popular choices since it does not suffer from the vanishing gradient problem. The MLPs often becomes more sparse because it sets some features to zero. It is claimed that ReLU learns multiple times faster than both the hyberbolic tangent and the sigmoid function.

- Leaky ReLU function:

$$g(a) = \begin{cases} a & if\ a \geq 0 \\ \alpha \cdot a & otherwise \end{cases}$$

The fact that ReLU can have some hidden covariates that is zero which can be an advantage. A disadvantage of ReLU is some neurons may die out if the neurons are mapped to zero. The Leaky ReLU is designed to give such neurons a chance to get back into action, but not too easily, so $\alpha > 0$ is chosen small (typical $\alpha = 0.01$)

- ELU - exponential linear unit:

$$g(a) = \begin{cases} a & if\ a \geq 0 \\ \alpha(\exp(a) - 1) \cdot x & otherwise \end{cases}$$

Like the Leaky ReLU the ELU is designed to avoid the dead ReLU problem.

With a understanding of a single neuron we continue to the architecture of a MLPs.

4.2.2 Architecture Of MLPs

A MLPs consists of a input layer, where all R features enter, L hidden layers, and an output layer. Each hidden layer and the output layer consists of multiple neurons, where the width of the layer is the number of neurons in that layer (m^l) (see figure 4.2). The networks inputs are called the input layer, the output layer is the output of the neural network. The layers between the input and output layer are hidden layers. This could be an explanation why the field is called Deep learning, because of a deep structure of layers. In each hidden layer a linear combination of the features from the previous layer is made and then an activation function is applied in order to create the new hidden features in that layer (see figure 4.2). The output in MLPs is a larged nested function, where the input layer go through a chain of functions until reaching the output.

$$\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{f}_1 \circ \boldsymbol{f}_2 \circ \cdots \circ \boldsymbol{f}_{L+1} \tag{4.2}$$

$$where\ \boldsymbol{f}_i : \mathbf{R}^{m^{i-1}} \to \mathbf{R}^{m^i} \quad i = 1, \ldots, L+1 \tag{4.3}$$

Each function in the composition of functions corresponds to a layer of neurons.

$$\boldsymbol{f}_i(x) = \boldsymbol{g}(\mathbf{W}^T \boldsymbol{x} + \boldsymbol{w_0}) \quad x \in \mathbf{R}^{m^{i-1}}\ and\ \mathbf{W} \in \mathbf{R}^{\boldsymbol{m^{i-1}} \times \boldsymbol{m^i}} \tag{4.4}$$

So the function maps a vector to a vector, the hidden layers will often be denoted $\boldsymbol{h}$ where for each single neuron, we map a vector to a scalar by:

$$h_i = g(\boldsymbol{x}^T \cdot \mathbf{W}_{:,i} + (w_0)_i)$$

A layer is a vector of neurons, hence the transformation from one layer to the next can be interpreted as multiple vector to skalar transformations, where each skalar/neuron acts in parallelle. The different view motivates the initial presentation of single neuron network (section 4.2.1).

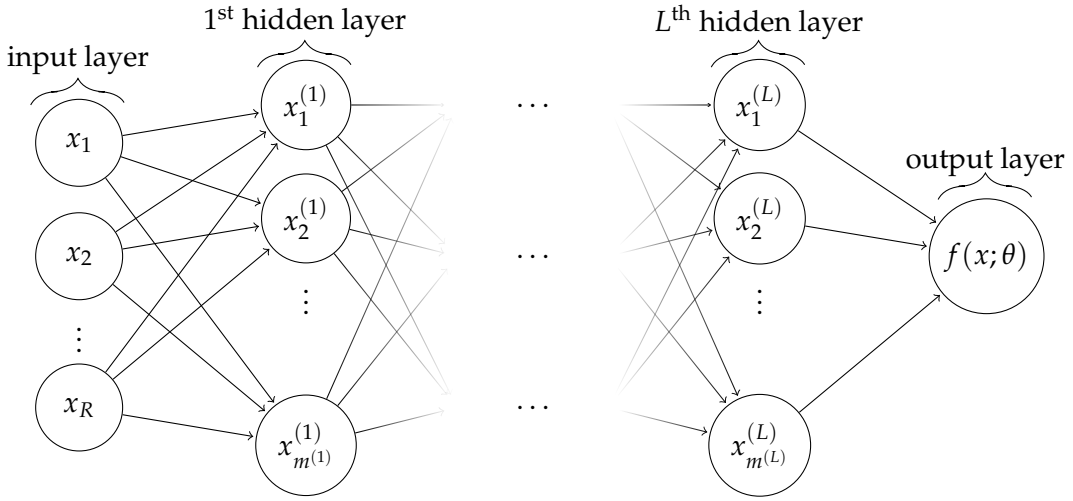

FIGURE 4.2: Multilayer perceptrons with $(L+1)$-layers with P input features and 1 output. The l^{th} hidden layer contains $m^{(l)}$ hidden neurons.

So the MLPs is not like classical linear regression in section 3.3 where a single linear transformation from input to output is applied. The unique attribute of neural network is the ability to approximate any kind of function[1], because of the flexiblility with applying multiple functions to the input layer. The neural network has a lot of different design options, where e.g. hidden layers, layer width, depth, activation functions etc. are hyperparameters. In general it is recommended to use many hidden covariates (neurons) rather than too few. The danger of overfitting is than avoided by introducing some kind of penalty to avoid that the model becomes overly large, which we will discuss further in section 4.2.4.

To fit a model the model need initialization of weights, biases and activation functions. In order to measure the performance of the model, we need a function to measure the difference between the approximation $f(\boldsymbol{x};\boldsymbol{\theta})$ and the target values $f^*(\boldsymbol{x})$. The function used to quantify this approximation is the loss function, where the cost function is the average over the loss functions. The cost function tells how close the predition $f(\boldsymbol{x};\boldsymbol{\theta})$ is to the target y. The cost function is key to improving our model or in machine learning lingo training the model, hence the next section will cover model training.

4.2.3 Training The Network

Training the network is key for building a high quality model. The performance of the model is measured by the cost function, where the cost function used in this

[1] Universal Apporximate Theorem page 194 (Goodfellow, Bengio, and Courville, 2016)

thesis is the the empirical risk function.

$$J(\theta) = E_{(\boldsymbol{x},y)\sim\hat{p}_{data}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y) = \frac{1}{k}\sum_{k=1}^{K} L(f(x_k;\theta), y_k)$$

The loss function L in empirical risk function for training is chosen to be quadratic, hence for training we penalize with mean square error (MSE) as our cost function.

From above we see that the cost function is a way to measure the approximation of $f^*(x)$ by our model $f(\boldsymbol{x};\boldsymbol{\theta})$. The training is important e.g. imagine after random initialization of parameter and construction of the MLPs we reported the output given the inputs of the model. This model would probably results in a high cost function value and bad performance because the given weights would produce a function that do not fit training data. The way out of the high cost function is to try to minimize the cost function over the weightspace hence training for MLPs is essentially a optimization of a nonconvex function. Remember the MLPs is a chain of functions (section 4.2.2) where the structure often makes the optimization a nonconvex problem hence a global minimum is seldom archived. Other pitfalls for optimization of MLPs are weight symmetry, steep cliffs, saddle points, vanishing and exploding gradient.

The actual optimization algorithms for MLPs are based on gradients, where we make small local moves. The overall goal is to find θ to reduce the test error. Within gradient methods there are batch gradient descend and minibatch stochastic gradient descend, where the former is training on the whole dataset, and the latter is only for a subset of the dataset. The minibatch methods have the advantage it can parallelized hence faster training. An epoch is the number of complete dataset training cycles to update the weights. For the minibatching techniques it is important to random sample from the whole dataset in order to get unbiased gradient estimation.

The goal of the optimization algorithms is to find critical points $\nabla J(\theta) = 0$, hence to obtain the critical points the iterative methods move in the opposite direction of sign of derivative $\nabla J(\theta)$. I.e. the gradient tells us how to update the parameters with a stepsize called the learning rate η:

$$\theta_{new} = \theta_{old} - \eta\nabla J(\theta_{old})$$

$J(\theta)$ is the cost function, hence remember we use the MSE:

$$J(\theta) = \frac{1}{K}\sum_{k=1}^{K} L(f(x_k;\theta), y_k) = \frac{1}{K}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2$$

Common method to estimate the parameters is gradient descent, stochastic gradient descent (SGD) and Adam, where all the optimization algorithms are iterative. Gradient descend uses the whole batch for each update, where Adam and SGD use minibatches for each update. The Adam algortihm use a adaptive learning rate, where the two others use constant or descending learning rate. The Adam method makes greater progress in more gently sloped directions of weightspace compared to SGD and gradient descent.

Besides choosing a optimization procedure the initialization of the parameters

are important. There is a lot different suggestions to initialize parameters, but there is no general golden rule at the moment because lack of understanding of the optimization procedure in neural nets. Often the practioners tend to use simple and heuristic methods where it has been shown the initialization breaks weight symmetry.

The most common way of finding gradients is the backpropagation algoritm, where the basic idea is the chain rule from calculus.

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

To understand backpropagation it is often useful with a computational graph created by forward propagation, where the backpropagation compute the derivative from output layer to input layer by going backward in the computational graph. The training process is a forward-backward algorithm. Different starting values of θ will result in different parameters. The good news is that these predictors typically do not differ by very much. It is recommended to work with a set of different starting values, and then use as a final predictor the average of the individual predictors stemming from each starting value.

4.2.4 Regularization

The number of parameters and the capacity of neural networks arise often the problem of overfitting, i.e. the model does not generalize well on new data. Regularization is a technique that constains our optimization problem to discourage complex models, hence avoid the problem of overfitting. Some common methods for deep learning are parameter norm regularization, early stopping and dropout.

Early stopping is a effective and simple method for regularization. Compared to parameter norm regularization the early stopping algorithm does not harm the learning dynamics. The early stopping method is also computational efficient, hence it makes a popular regularization method for deep learning. The idea of early stopping is that the iterative training algorihm keeps improving the train error, but training too extensively leads the test error to rise. The idea is then to split your data into validation and train datasets, where you determine the best $\hat{\theta}$ and the corresponding training steps $\hat{i}$ by iterative comparing the cost function on the validation set. The algorithm stops after a predefined number of steps without improving the cost function on the validation set. The number of epochs to wait for the cost function not to improve is called the patient for the early stopping algorithm.

Like early stopping the dropout method is computationally inexpensive. The idea is to remove neuron randomly at each training loop, i.e. when updating the gradient, each node is kept with probability p, independently of each other. To perform dropout a binary mask is sampled independently for each iterations, where the probability p for 1 is another hyperparameter. The goal is to minimize $E_{\boldsymbol{\mu}} J(\boldsymbol{\theta}, \boldsymbol{\mu})$ where $\boldsymbol{\mu}$ is the mask. The result of the procedure is more robust features and a regularizing effect on most models.

There is a lot of design options for deep learning, where the choices should be specific for the given task. There is a no free lunch theorem for machine learning, which says no model is superior for all tasks (page 114 (Goodfellow, Bengio, and

Courville, 2016)). The design for specific tasks are important where training error and test error can be improved by designing the model for the given task. Another aspekt is the computational resources i.e. the memory space and computational time.

Pricing derivatives with deep learning methods have two clear benefits. The first is computational time, where after a model is trained, then the model is far superior to the methods presented in chapter 3. Another advantage is the non-linearities of the model making it possible to fit more complex functions. The application of deep learning in option pricing theory will be explored in next chapter.

Chapter 5

Option Pricing And Deep Learning

Deep learning can be applied to option valuation in different ways. We will investigate to use MLPs regression in the LSM algorithm instead of the linear model. The two methods will be compared numerically to numerous of options. The second use of neural network is to simulate the input variables and use existing methods to value the option i.e. finding the target values. The task is then to learn the pricing formula from the training set $(\mathbf{X}, \boldsymbol{y})$. Both methods fall within supervised regression where we will use MLPs network introduced in section 4.2 to approximate the mappings. The risky assets are modelled with black scholes theory from earlier chapters hence the approapiate simulating methods are already presented (Chapter 2, 3). The theory in chapter 4 will be specialized for the specific task and discussed. The advantage of MLPs is that the model scale well to high dimensional data, where e.g. polynomial regression (section 3.3) is prone to overfit and slow compared to MLPs for high dimensional tasks.

5.1 Multilayer Perceptrons Regression For Optimal Stopping

The first application of neural network is to investigate, if the LSM method can be improved by using MLPs regression to approximate continuation value instead of using the linear model in LSM. This section is influenced by (Longstaff and Schwartz, 2001; Lapeyre and Lelong, 2019; Kohler, Krzyżak, and Todorovic, 2010), where the approximate scheme with only in-the-money paths (ITM) is inspired by (Longstaff and Schwartz, 2001) and the idea of using neural network instead of the linear model comes from (Kohler, Krzyżak, and Todorovic, 2010; Lapeyre and Lelong, 2019). The algorithm is based on approximating the american put option with one or several underlyings by considering a bermudan option, which converge to the american option by making the equidistant time-steps sufficiently small.

The setup is the same as for the LSM presented in chapter 3, where the difference is the regression to estimate the expected continuation value. We will use the theory of deep learning more specifically MLPs to purpose an alternative to the linear model. The main difference lay in the capacity of the two models, where MLPs through its activation functions can capture non linearty which classical linear regression cannot. Another advantage is that deep learning is known for breaking the curse of dimensionality, hence the MLPs could be better suited for multivariate contingent claims than the classical LSM. The method will sometimes be referred to **MLPs I**.

5.1.1 Recap MLPs

We model the MLPs with a non linear function

$$s \in S \in \mathbb{R}^R \mapsto \Psi(s;\theta) \in \mathbb{R}$$

where Ψ is the function decomposition and given by

$$\Psi = a_{L+1} \circ g_L \circ a_L \circ \cdots \circ g_1 \circ a_1 \quad \textit{where } L \in \mathbb{N}$$

We refer to chapter 4 for notation, where $a_l(x) = \mathbf{W}_l^T \boldsymbol{x} + \boldsymbol{w}_{0,l}$. We embed all the parameters of different layers into a unique high dimensional parameter

$$\theta = (\mathbf{W}_l, \boldsymbol{w}_{0,l})_{l=1,\dots,L+1} \in \mathbb{R}^{N_d} \text{ with } N_d = \sum_{l=1}^{L+1} m^{(l)}(1 + m^{(l-1)})$$

. We will restrict our parameter space by a increasing sequence $(\gamma_p)_{p\in\mathbb{N}}$ such that $\lim_{p\to\infty} \gamma_p = \infty$ and $p \in \mathbb{N}$ is the maximum number of neuron in each hidden layer. We define the set:

$$\Theta_p = \{\theta \in \mathbb{R} \times \mathbb{R}^p \times (\mathbb{R}^p \times \mathbb{R}^{p\times p})^{L-1} \times \mathbb{R}^p \times \mathbb{R}^{p\times p} : |\theta| \leq \gamma_p\}$$

By the definition of the set we restrict our MLPs to the set:

$$\mathcal{NN}_p = \{\Psi(\cdot;\theta) : \theta \in \Theta_p\}$$

Unfortunately the $\mathcal{NN}_p$ is not a vector space, hence the regression cannot be interpreted as an orthogonal projection as for the LSM. The MLPs method is justified by the "Universal Approximate Theorem" (theorem 5.1)

Theorem 5.1. ***Universal Approximation Theorem*** *Assume that the activation function g is nonconstant and bounded. Let μ denote the probability measure on $\mathbb{R}^r$, then for any $L \geq 1$, then $\mathcal{NN}_\infty$ is dense in $L^2(\mathbb{R}^r, \mu)$*
where $\mathcal{NN}_\infty = \cup_{p\in\mathbb{N}} \mathcal{NN}_\infty$ (p. 4 (Lapeyre and Lelong, 2019))

Note that the above theorem can be rephrase in terms of approximating random variables

Remark 5.2. *Let Y be a real valued random variable suct that $E[Y^2] < \infty$. Let X be a random variable taking values in R^r and $\mathcal{G}$ be the smallest σ-algebra such that X is $\mathcal{G}$ measurable. Then, there exists a sequance $(\theta_p)_{p\geq 2} \in \prod_{p=2}^{\infty} \Theta_p$ such that $E[|Y - \Psi_p(X;\theta_p)|^2] = 0$. Therefore, if for every $p \geq 2$, $\alpha_p \in \Theta_p$ solves*

$$\inf_{\theta\in\Theta_p} E[|\Psi_p(X;\theta) - Y|^2]$$

Then the sequence $((\Psi_p(X;\alpha_p))_{p\geq 2})$ converge to $E[Y|X]$ in $L^2(\Omega)$ when $p \to \infty$ (p. 5 (Lapeyre and Lelong, 2019)).

The remarks reveals that the MLPs can be used to approximate the conditional expectation instead of a linear function of the elements of the basis.

5.1.2 The Algorithm

The algorithm is similar to the LSM, but the regression step is slightly different. We will use the same assumptions as in LSM section and same principles. Recall the

optimal stopping problem can be solved with dynamic programming principle on the optimal policy:

$$\begin{cases} \hat{\tau}_{t_N}^{k,p,K} = t_N \\ \hat{\tau}_{t_n}^{k,p,K} = t_n \cdot 1_{\{G(S^{(k)}(t_n)) \geq \Psi_p(S^{(k)}(t_n);\hat{\theta}_{t_n}^{p,K})\}} + \hat{\tau}_{t_{n+1}}^{k,p,K} \cdot 1_{\{G(S^{(k)}(t_n)) \geq \Psi_p(S^{(k)}(t_n);\hat{\theta}_{t_n}^{p,K})\}} \end{cases} \quad for\ n = 0, \dots, N-1 \tag{5.1}$$

Where the estimator $\hat{\theta}_{t_n}^{p,M}$ is given by minimizing squared sample distance of the estimated continuation value and the realized continuation value:

$$\hat{\theta}_{t_n}^{p,M} = \underset{\theta \in \Theta_p}{\operatorname{argmin}} \frac{1}{K} \sum_{k=1}^{K} \left(G(S^{(k)}(\tau_{t_{n+1}}^{k,p,K})) - \Psi_p(S^{(k)}(t_n);\theta) \right)^2$$

Finally in analog with the LSM section the approximated time 0 price for the option is

$$U^{p,K}(0) = \max\{G(S(0)), \frac{1}{K} \sum_{k=1}^{K} G(S^{(k)}(\hat{\tau}_{t_1}^{k,p,K}))]\} \tag{5.2}$$

5.1.3 Convergence

The MLPs regression method enjoy the same convergence results presented for the LSM algortihm, i.e.

Theorem 5.3. *Assume that*

$$E[\max_{0 \leq t_n \leq T} |G(S(t_n))|^2] < \infty$$

. Then $\lim_{p \to \infty} E^Q[G(S(\tau_{t_n}^p))|\mathcal{F}_{t_n}] = E[G(\tau_{t_n})|\mathcal{F}_{t_n}]$ *in* $L^2(\Omega)$ *for all* $n \in \{1, 2, \dots, N\}$
Proof p. 7-8 (Lapeyre and Lelong, 2019)

The above theorem states convergence of the neural network approximation, i.e. $U^p(0) \to U(0)$ which is similar to the convergence result for LSM.

Theorem 5.4. ***Strong law of large numbers:*** *Assume*

A1: For every $p \in \mathbb{N}$, $p > 1$, *there exist* $q \geq 1$ *and* $\kappa_p > 0$ *such that*

$$\forall s \in \mathbb{R}^R, \forall \theta \in \Theta_p, \quad |\Psi_p(s,\theta)| \leq \kappa_p(1 + |s|^q)$$

Moreover $\forall n \in \{1, 2, \dots, N-1\}$, *a.s. the random functions* $\theta \in \Theta_p \mapsto \Psi_p(S(t_n), \theta)$ *are continuous. Note* Θ_p *is a compact set, hence the continuity is uniform.*

A2: For q defined in A1, $E^Q[|S(t_n)|^{2q}] < \infty \quad \forall n \in \mathbb{N} \cup 0$

A3: $\forall p \in \mathbb{N}$, $p > 1$ *and* $\forall n \in \{1, 2, \dots, N-1\}$,

$$P(S(t_n) = \Psi_p(S(t_n); \theta_{t_n}^p) = 0$$

A4: $\forall p \in \mathbb{N}$, $p > 1$ *and* $\forall n \in \{1, 2, \dots, N-1\}$, *if* θ^1 *and* θ^2 *solves*

$$\underset{\theta \in \Theta_p}{\operatorname{argmin}}\, E^Q[|\Psi_p(S(t_n);\theta) - S(\tau_{t_{n+1}}^p)|^2]$$

then $\Psi_p(s, \theta^1) = \Psi_p(s, \theta^2)$ *for almost all x.*

If A1-A4 hold, then for $\zeta \in \{1,2\}$ *and every* $n \in \{1,2,\dots,N\}$

$$\lim_{K\to\infty} \frac{1}{K} \sum_{k=1}^{K} \left(G(S(\hat{\tau}_{t_n}^{k,p,K})) \right)^{\zeta} = E[\left(G(S(\tau_{t_n}^{p})) \right)^{\zeta}] \quad a.s. \tag{5.3}$$

Proof p. 13-14 (Lapeyre and Lelong, 2019)

The last result is the strong law of large numbers for the value function, i.e. $U^{p,K}(0) \to U^{p}(0)$ *a.s.* $\quad for\ K \to \infty$

5.2 Multilayer Perceptrons Regression Pricing From Existing Methods

The MLPs pricing method from existing methods has a different approach than the other methods, because it is only data drivin. The MLPs could easily be used to real data, which is investigated in (Gaspar, Lopes, and Sequeira, 2020). We revisit the work from (Hirsa, Karatas, and Oskoui, 2019), where we try to extend the pricing models to options with two underlying risky assets. The model will be the classical GBM model presented in earlier chapters. By choosing the GBM model the MLPs pricing method is ready for investigation both for vanilla options and exotic options. We stress that the MLPs pricing method is not restricted to GBM assumption, but can be applied to other models such as Hesten, Variance Gamma and real market data. The advantages with MLPs is the fast parameter estimation compared to classical methods binomial lattice and LSM once trained. With the increased speed for pricing it can cost accurary specially if the data is sparse, which can arise when using the method for exotic options on real market data. For practical application the accuracy is severe if the predicted price is not within the bid-ask spread, because the aim is to price fast without introducing arbitrage. We will present results for european call, american put and american put miminum on two assets options presented in earliar chapters with MLPs pricing methods.

For deep learning the hyperparameters is important for finding the right model for pricing, where different choices will be empirical presented under training. For the given task polynomial regression can also be used, but we will see later why MLPs regression is preferred. The section is split into three sections "Data", "Training" and "Performance".

5.2.1 Data

The generation of labels are the computational expensive part of the MLPs method, because the method needs enough samples to approximate the function f^* well. The upside after generation of labels is that the method is computationally fast and easy to implement with basic knowledge of deep learning[1]. The labels will be generated by existing methods presented in chapter 3, where the input parameters will be uniform sampled or quasi sampled with halton sequences.

For both european and the american univariate contingent claims the parameter in-sample will be the same, where we remember the 5 parameters for pricing an univariate contingent claims[2]. Note for simulation of labels that the european call and american put options are first order homogeneous function in $(S(0), K)$, hence the valuation formulas can be modified:

$$\frac{c(S(0),K)}{K} = c(\frac{S(0)}{K}, 1) \quad and \quad \frac{P(S(0),K)}{K} = P(\frac{S(0)}{K}, 1)$$

The alternative representation above reduces the number of parameters needed for simulation to 4, because instead of simulate both S and K, moneyness $(\frac{S(0)}{K})$ is only simulated. This property is not shared for the american bivariate contingent claim. For the european call and american put the input matrix $\mathbf{X}$ is different combinations

[1] Chapter 4
[2] E.g. the european call proposition 2.16

of the 4 parameters within the parameter range (table 5.1), where it is assumed the number of trading days are 252 in a year. The parameter ranges are risk free rate of return between 1-3 %, maturity between 1 day to 3 years, moneyness between 0.8 and 1.2 and volatility between 0.05 and 0.5.

TABLE 5.1: Parameter ranges for american put and european call options

Derivative	**r**	**T**	**Moneyness**	σ
Euro. Call	1%-3%	1d - 3y	0.8-1.2	0.05-0.5
Amer. Put	1%-3%	1d - 3y	0.8-1.2	0.05-0.5

The american put minimum option for two underlyings will require additional parameters, because now we have two spots, two volatilities and correlation between the assets. The first order homegeneity does not hold for multivariate contingent claims, hence the strike and the two spots are also needed. The parameters considered for the bivariate contingent claims are T, r, K, $S_1(0)$, $S_2(0)$, σ_1, σ_2 and ρ. So a total of 8 parameters and the given parameters ranges are given in table 5.2, where we reduced the number of combinations by setting the two spots constant to 100 and let the strike be the same for both options.

TABLE 5.2: Parameter ranges for exotic european put and american put options

Derivative	**r**	**T**	K	$S_1(0)$	$S_2(0)$	σ_1	σ_2	ρ
Euro. Put Min.	1%-3%	1d - 3y	80-120	100	100	0.05-0.5	0.05-0.5	0.05-0.5
Amer. Put Min.	1%-3%	1d - 3y	80-120	100	100	0.05-0.5	0.05-0.5	0.05-0.5

The simulationen of the parameter ranges for the training dataset is done by quasi-random Halton sequences sampling to obtain lower discrepancy, where the test sets are sampled with random uniform sampling. The quasi random sampling differ from the ordinary random sampling since they make no attempt to mimic randomness. The aim for quasi sampling is to increase accuracy specifically by generating points that are too evenly distributed to be random. The halton sequence and uniform sampling generates points between 0 and 1, hence we need to apply a transformation to get the parameter ranges:

$$r \cdot (range\ of\ parameter) + lowerBound \quad where\ r = halton\ point$$

After the simulation of combinations of the input parameters within the given ranges the labels can be generated from existing methods. For the european call option the generation of labels is done by the classical B-S formula for call options given in proposition 2.16. The B-S pricing formula is well known and has an analytical solution, hence it is relatively fast to generate labels in this model. The american options require numerical methods, hence the generation of labels are more computational expensive. The labels for the american options are generated by CRR for american put option with 1 underlying stock and by BEG for two underlyings with 100

equidistant time-steps presented in chapter 3. When the datasets ($\mathbf{X}$, $\boldsymbol{y}$) are generated we are left with a regression task. The regression task is to predict the price $\boldsymbol{y}$ from the input parameters $\mathbf{X}$.

The datasets generated for within the parameter ranges will be referred to as the in-sample data set. The training dataset is a in-sample dataset, which is split up into a training and validation dataset in order to approximate model performance on the test sets. The training dataset is used to update the parameters internal in the model e.g. weights, biases, et cetera and finetune hyperparameters for choosing the optimal model design. To avoid overfitting and good generalization models the validation set is useful, because the trained model has not seen the data before evaluation on the validation set. The validation set is randomly subsampled from the training set, where the validation dataset constitute 20 procent of the training set. To check the robustness of the regression we choose to sample two test out-of-sample datasets, which means we simulate data with modified parameter ranges.

The test datasets out-of-sample is simulated by uniform sampling adjusting the parameter range for either moneyness or maturity for the options (table **??**). The test dataset has not been seen by the model in the training process, hence we get a unbiased evaluation of the model. The aim with producing different test and validation data sets is to measure the models performance at interpolation and extrapolation. The dataset for the bivariate american contingent claim will be similar to the american put by only adjusting either the strike or the maturity.

TABLE 5.3: Parameter ranges for european call and american put option

Dataset	Derivative	**r**	**T**	**Moneyness**	σ
In-Sample	Euro. Call	0.05-0.5	1d-3y	0.8-1.2	1%-3%
Out-Of-Money		0.05-0.5	1d-3y	0.6-0.8	1%-3%
Longer Maturity		0.05-0.5	3y-5y	0.8-1.2	1%-3%
In-Sample	Amer. Put	0.05-0.5	1d-3y	0.8-1.2	1%-3%
In-The-Money		0.05-0.5	1d-3y	0.6-0.8	1%-3%
Longer Maturity		0.05-0.5	3y-5y	0.8-1.2	1%-3%

By above discussion we simulate a training dataset, where the simulation for the american put is visualized in figure 5.1. The marginal distributions shown is for 300.000 data samples ($\mathbf{X}, y$) generated by halton sequences and the CRR model, where the marginal distributions for the features cover the parameter range almost uniform and the simulated y lies with most values at zero and maximum at 0.387 rounded to three decimals. The marginal distributions shows that we have succesfully generated parameters in the given ranges and the parameters are evenly spaced in the ranges. In the model performance section the out-of-sample and in-sample test datasets will be used to check the extrapolation and interpolation of the models. The test datasets are 60.000 generated data points with uniform sampling and the parameter ranges for the options with 1 underlying is given in table 5.3.

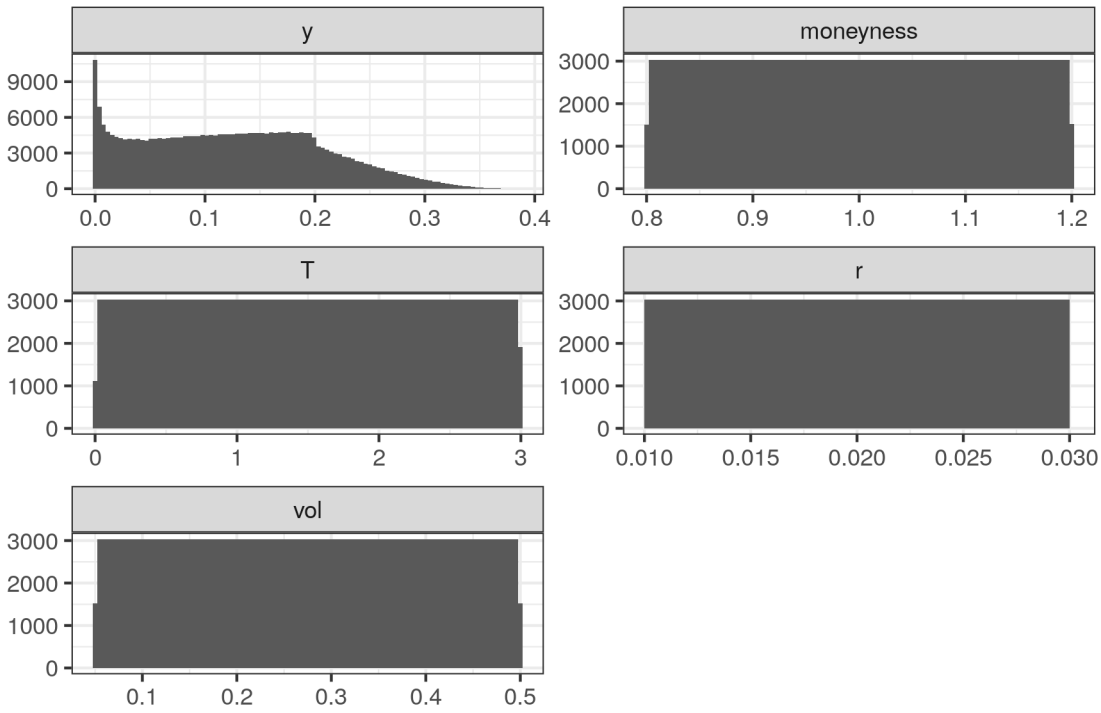

FIGURE 5.1: Quasi random simulation with halton sequence for input variables and CRR for generation of labels for american put option.

5.2.2 Training

The aim for training is that the model will learn the pricing function f^*. Once the dataset $(\mathbf{X}, \boldsymbol{y})$ is generated the model can be trained to approximate the true function f^*. To train the model we use MLPs regression to infer the pricing function from the generated data, hence the approach is model free, because the data is only needed. We will also present a standard linear model to compare the MLPs with the classical regression methods. We want to have a fast, robust and accurate model after training on the training set. To train the model we need a measure for the error, where the standard mean squared error (MSE) for regression is applied. I.e. the cost function is chosen to be the empirical risk function with a quadratic loss function:

$$J(\theta) = \frac{1}{K} \sum_{k=1}^{K} (y_k - \hat{y}_k)^2$$

The MSE penalizes outliers stronger than e.g. mean absolute error (MAE), but it penalizes small deviations less. To avoid overfitting we regularize using early stopping with a patient of 5 epochs, but with a maximum of 100 epochs for computational reasons. To update the weights the Adam optimization algorithm is chosen with learning rate η found by hyperparameter tuning.

5.2.2.1 Hyperparameter Tuning

A research area within deep learning is to tune the hyperparameters to the specific task, where both manually search and automated search can be used. To choose the best hyperparameters there is several choices, where the most basic automated task is random search and grid search. The random search is to define a predefined range to pick randomly from. This method can be effective to discover new hyperparameter values or combination, but it can be considerably more computational expensive than the grid search. The grid search is to search in a grid with each hyperparameters in each dimension, this kind of hyperparameter search has fast computational time relative to the random search, but requires some expert knowledge about the hyperparameters.

We test empirically the best hyperparameters for the MLPs, where the validation set is used. For hyperparameter tuning grid search is conducted to find the optimal set of hyperparameters where we will look at dataset size, learning rate and batch-size.

Firstly the european call option regression is investigated, where we vary the dataset size and batch size. For the Adam optimization algorithm we choose a learning rate $\eta = 0.001$. The goal is to quantify how large a dataset(d) and batch size(b) are needed for having a high quality model for the european option. The datasets for an european call option considered are in-sample datasets of size 1K, 10K, 100K, 300k and 1M data samples, where the validation set is subsampled from the training data set. The batch sizes are 64, 256 and 1024, i.e. we have the combinations expressed by the cartesian product:

$$b \times d = \{(b,d) : b \in \{64, 256, 1024\} \; and \; d \in \{1K, 10K, 100K, 300K, 1M\}\}$$

By inspiration from (Hirsa, Karatas, and Oskoui, 2019) we train a MLPs model with 4 layers, 120 neurons in each hidden layer and 1 neuron in the output layer. In each layer we choose the activation function leaky ReLU, which is one of the most popular choices for activation functions. The number of data samples is relevant for real data, because for real market data there is not unlimited market data available.

Table 5.4 shows that the the model performs very well for in-sample data with 10K-300K datapoints, hence the biggest dataset with 1M seems not to be worth the computational cost. The model is only trained once on each datasets, so there is some randomness on each run, but the picture is clear. The model to interpolate prices for european call options in-sample data does not significant improve with gathering more data than 10K-300K datapoints, which is good news for using the method on real market data. Beaware that the simulated data can underestimate the validation error, because we have a controlled setup where the parameter range is within a predetermined range. For the controlled setup with simulated data we can choose arbitrary many datapoints albeit making the method more computational expensive. By weighting both the computational cost and accurary, we choose to work with 300K datapoints and a batch size of 64 for the european option.

The european option needs fewer parameters compared to the american put on minimum of two assets, hence the dataset might need to be larger for that study. We conduct a large grid search with varying the learning rate η, the batchsize b and the

TABLE 5.4: Hyperparameter tuning of dataset size and batch size for the american put bivariate contingent claim. The table shows validtion loss in ascending order for different hyperparameter combinations and for the interested reader the tensorboard is online (link tensorboard 1)

Dataset Size	Batch Size	Training Loss	Validation Loss	Time: min:sec
1M	256	$8.094 \cdot 10^{-7}$	$8.7764 \cdot 10^{-7}$	3:38
300K	64	$7.0562 \cdot 10^{-7}$	$1.0849 \cdot 10^{-6}$	2:49
1M	1024	$1.1596 \cdot 10^{-6}$	$1.1578 \cdot 10^{-6}$	5:58
300K	256	$9.579 \cdot 10^{-7}$	$1.3268 \cdot 10^{-6}$	2:29
300K	1024	$1.5367 \cdot 10^{-6}$	$1.4138 \cdot 10^{-6}$	9:28
1M	64	$3.4919 \cdot 10^{-7}$	$1.9197 \cdot 10^{-6}$	8:24
100K	256	$2.243 \cdot 10^{-6}$	$2.1192 \cdot 10^{-6}$	1:02
100K	64	$1.9565 \cdot 10^{-6}$	$2.5738 \cdot 10^{-6}$	1:01
100K	1024	$3.22 \cdot 10^{-6}$	$4.4754 \cdot 10^{-6}$	2:00
10K	256	$1.1179 \cdot 10^{-5}$	$1.0980 \cdot 10^{-5}$	0:37
10K	64	$1.0043 \cdot 10^{-5}$	$1.9830 \cdot 10^{-5}$	0:15
1K	64	$6.1389 \cdot 10^{-5}$	$7.8711 \cdot 10^{-5}$	0:22
10K	1024	$8.7067 \cdot 10^{-5}$	$8.1122 \cdot 10^{-5}$	0:32
1K	256	$1.2032 \cdot 10^{-4}$	$1.2504 \cdot 10^{-4}$	0:20
1K	1024	$7.5948 \cdot 10^{-3}$	$7.3595 \cdot 10^{-3}$	0:08

dataset size d, where the grid of the cartesian product of η, b and d is searched.

$$\eta \times b \times d = \{(\eta, b, d) : \eta \in \{0.0001, 0.001, 0.01\},\ b \in \{8, 64, 256, 512, 1024\}\ and\ d \in \{1K, 100K, 300K\}\}$$

Besides varying the hyperparameters for η, b and d the other hyperparameters is the same values as for the european call option.

Looking at table 5.5 the best configuration for both the validation loss and training loss is for $(\eta = 0.0001, b = 8, d = 300K)$, but the computational cost is by far the most expensive by taking an hour for training with these hyperparameters. The reason lay in the lowest batch size, biggest dataset and the smallest learning rate considered. The learning rate effect the computational speed, because we are training with early stopping regularization. Overall the dataset with 300K performed best for training error, which is expected, since a bigger dataset gives more learning opportunities. The batch size effects heavily the computational speed, since a high batch size gives fewer iterations per epoch. Considering computational cost and the loss the 300K data sets, the learning rate $\eta = 0.001$ and a batch size of 64 or 256 are preferred.

Hyperparameter tuning is highly computational expensive and there is many opportunities to test out. We have seem with grid search the optimal choice for the learning rate, batch size and datasize for the european call option and american put on minimum for two assets. The american put option uses the same parameters as the european option, hence we choose the same hyperparameters for the univariate contingent claims. Below we will try a different model than MLPs to see if we really need deep learning for this method.

TABLE 5.5: Hyperparameter tuning of dataset size, learning rate and batch size for the american put bivariate contingent claim. The table shows the top 10 best performing combinations for the training loss and for the interested reader the tensorboard is online (link tensorboard 2)

Dataset Size	Learning Rate	Batch Size	Train Loss	Val. Loss	Time: min:sec
300K	0.0001	8	0.0151	$3.615 \cdot 10^{-3}$	60:41
300K	0.001	64	0.0355	$6.676 \cdot 10^{-3}$	11:04
100K	0.0001	8	0.0649	0.0473	20:44
100K	0.001	8	0.0721	0.0373	21:41
300K	0.0001	64	0.0760	0.1548	13:51
300K	0.001	8	0.1046	8.8420	26:41
300K	0.01	256	0.4800	0.1400	2:37
300K	0.0001	256	1.0409	0.9533	6:40
300K	0.001	256	1.0681	0.9156	2:27
300K	0.001	512	1.0894	0.9345	3:11

5.2.2.2 Polynomial Regression

To compare MLPs and Polynomial regression the dataset and the performance metrics are the same, but the model training is obvously different. The dataset chosen is the dataset with 300.000 samples and the metric is MSE. For simplicity the european call option is investigated, where we fit polynomials up to degree 6 for comparision of the model capacity and fit.

$$y_i = \beta_0 + \beta_1 \cdot x_i + \cdots + \beta_n \cdot x_i^n + \epsilon_i \quad \textit{where } n = 1, 2, \ldots, 6$$

Plotting the fit actual vs predicted target variable (figure 5.2) it is clear, that the in-sample fit improves with the increased model capacity. The linear regression is too simple for pricing european option, but it looks like the 6 order polynomial actually performs better than the MLPs in the in-sample validation set table 5.6. Keep in mind that we do not only want good interpolation, but we also want good extrapolation for our fitted model.The out-of-sample data will reveal if the high order polynomial or MLPs have overfitted the data.

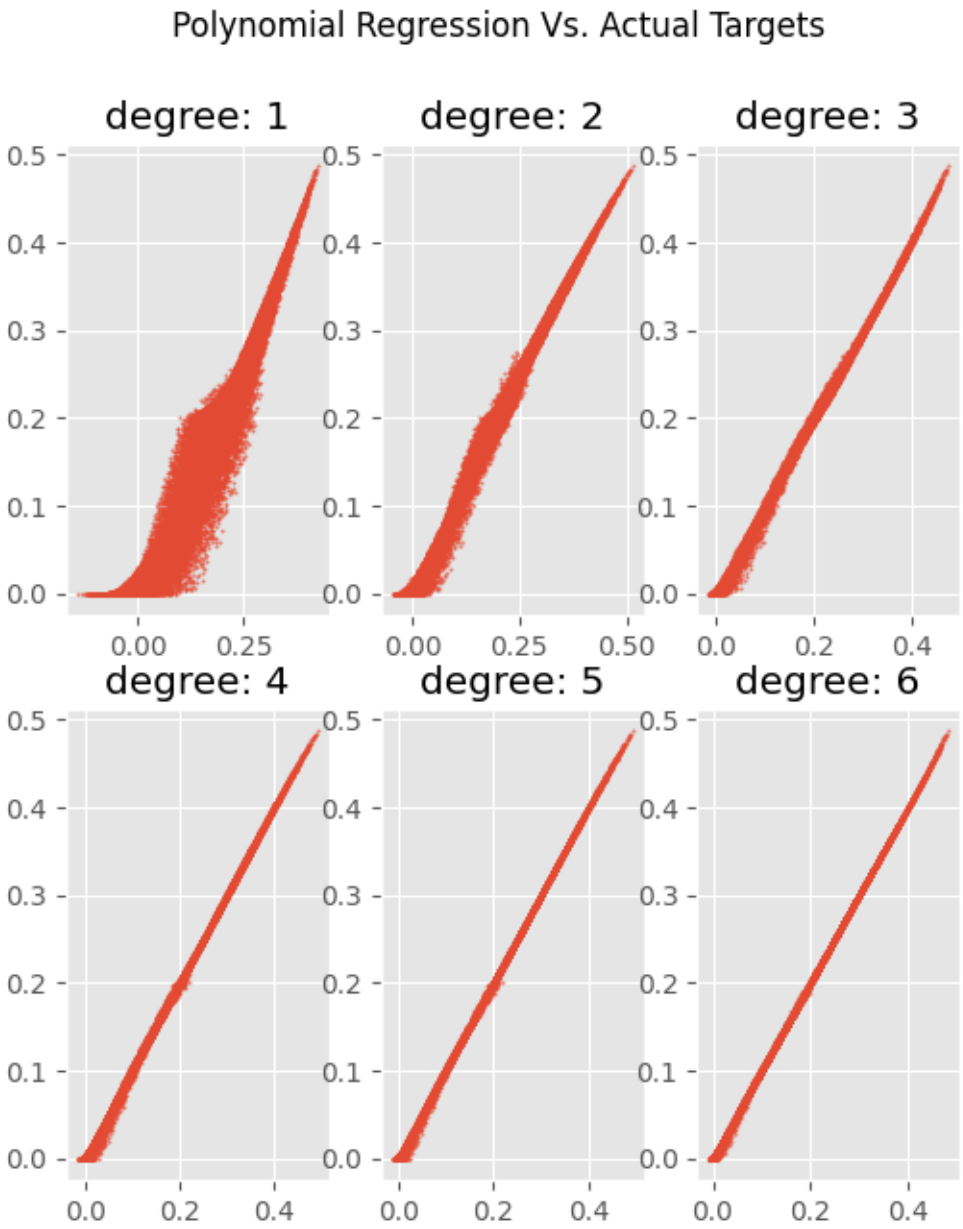

FIGURE 5.2: Predicted price based on polynomial regression of varying degree

Table 5.6 is created to compare the performance for each model. The table confirms that the linear regression has a worse fit than the other models with higher capacity. The difference on the MLPs and best performing polynomial model are less than $1 \cdot 10^{-6}$. The difference is negible so the fit for MLPs and polynomial regression of degree 4-6 performs all very well on the in-sample training data in terms of the validation loss.

TABLE 5.6: In-sample validation error for polynomial regression and MLPs on european call option

Model	Valalidation Loss
Linear Reg.	0.000631
2. degree poly.	0.000069
3. degree poly	0.000013
4. degree poly.	0.000004
5. degree poly.	0.000002
6. degree poly.	0.000001
MLPs	0.000003

5.2.3 Performance

The model performance is evaluated by MSE, RMSE, MAE and coefficient of determination, where all the measures evaluate how close the model predictions are with the actual targets. The first three measures ranges are $\mathbb{R}^+$, where the goal is to have the lowest value possible. For MSE close to 0 means that the model predictions does not differ a lot from the observed targets. The RMSE and MAE are same kind of measure, but the deviation is measured slightly different. The coefficient of determination has range $(-\infty, 1]$, where a higher value indicate a better model. Coefficient of determination provides a measure of how well observed targets are predicted by the model, based on the proportion of total variation of targets explained by the model.

5.2.3.1 European Call Option

The european call option is trained with the algortihm and hyperparameters described in the training section (section 5.2.2). By the hyperparameter investigation we choose a batch size of 64, learning rate $\eta = 0.001$ and a dataset of 300K samples. We compare the MLPs regression with the polynomial regression. Table 5.7 shows that the MLPs is superior at extrapolating, because the MLPs performs better on all metrics on the out-of-sample test datasets compared to fitted polynomial regressions with a degree between 1 and 6. For the in-sample test dataset the polynomial regression of order 6 and MLPs perform almost equally well, but the polynomial regression of 6. degree is a classical example of overfitting. The in-sample test data is similar to the in-sample training data, hence we expect the same magnitude of error for the test set as for the training set for in-sample testing. The performance measures show that the polynomial regression that was performing well on the In-Sample dataset was due to overfitting, because the high order polynomial regression does perform poorly on out-of-sample data (table 5.7). For the 6. order polynomial regression, we see a negative coefficient of determination, which means the model performs worse than the model with the mean as a horizontal line. This means the 6. degree polynomial have bad performance for out-of-sample data.

The MLPs has high predictive strength compared to the polynomials, because it performs well also on out-of-sample dataset. The MLPs show that it predicts out-of-money call option with less than a MSE with value 10^{-5}, where the fit for the long maturity test set has a higher MSE. The european options are liquid in the markets,

TABLE 5.7: Performance comparision of MLPs and polynomial regression on european call option. Shown the best performing regressions in the linear model and the worst performing in terms of MSE for in-sample and out-of-sample datasets

Model	Dataset	MSE	RMSE	MAE	Coefficent of Determination
MLPs	In-Sample	0.000000	0.000629	0.000486	0.999961
	Out-of-Money	0.000007	0.002644	0.001551	0.995911
	Long Matuiry	0.000197	0.014048	0.010061	0.986518
6. degree poly.	In-Sample	0.000001	0.000958	0.000591	0.999909
1. degree poly.		0.000636	0.025212	0.018326	0.936628
2. degree poly.	Long Maturity	0.001196	0.034577	0.026287	0.918316
6. degree poly.		0.043361	0.208233	0.111190	-1.962442
2. degree poly.	Out-Of-Money	0.000767	0.027694	0.022203	0.551246
1. degree poly.		0.005772	0.075973	0.060936	-2.377251

hence the pricing method can easily be trained on real data. The MLPs fit for in-sample data on the european call option is visualized in figure 5.3, which shows $\frac{c(S_0,K)}{K}$ predicted from the model and observed target values $\boldsymbol{y}$ divided by the strike. The figure shows a overall close fit. For the remaining part of this section the MLPs regression will only be considered, because the polynomial regression has shown bad performance for out-of-sample data.

5.2.3.2 American Put Option

The american put option is priced with the same MLPs algorithm as for the european call. Table 5.8 shows once a again a good fit, hence, we believe, we have a high quality model for the american put option.

TABLE 5.8: MLPs Performance on American Put Option

Dataset	MSE	RMSE	MAE	R^2
In-Sample	0.000002	0.001562	0.001278	0.999634
Out-Of-Money	0.000012	0.003519	0.002290	0.995778
Longer Maturity	0.000193	0.013894	0.009213	0.980835

5.2.3.3 American Put On Minimum of Two Assets Option

The american put on minimum of two option has the double amount of parameters compared to the univariate contingent claims, hence the performance might give slightly higher MSE. The other potential problem is the lack of generality when training, because the american put on minimum of two option is not a first order homogeneous function. The error is then on the actual price for the option and not on the price relative to the strike. Table 5.9 shows a good fit for both in-the-money and longer maturity sample, but the MSE is higher as expected compared to the relative prices for the univariate contingent claims.

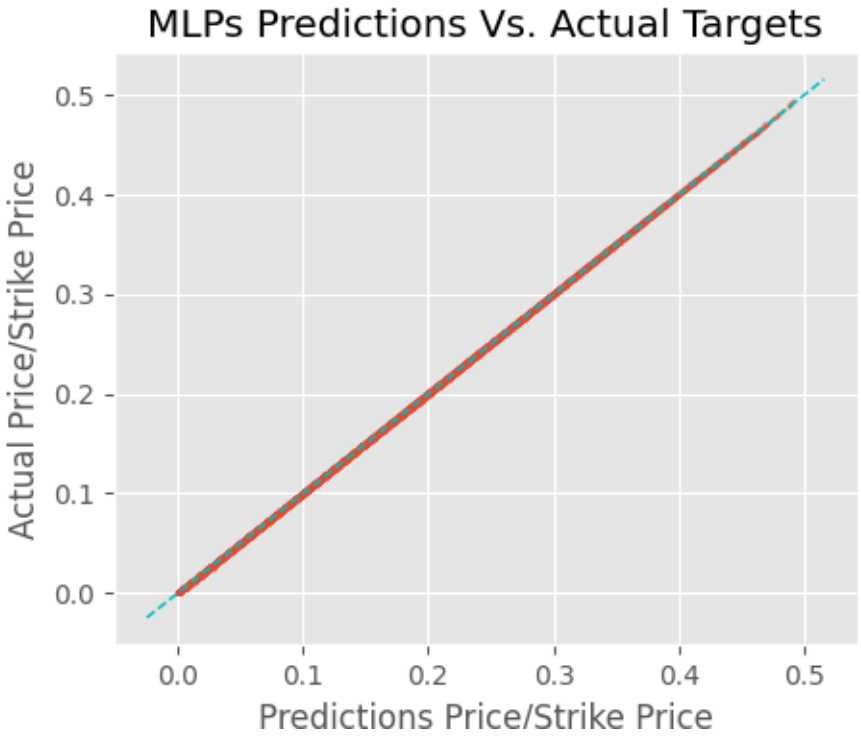

FIGURE 5.3: MLPs Performance on in-sample dataset on european call

TABLE 5.9: Performance of predictive strength for different regression models

Dataset	**MSE**	**RMSE**	**MAE**	R^2
In-Sample(TODO! Numbers)	0.099621	0.315644	0.001278	0.999634
In-The-Money	0.510718	0.714646	0.542326	0.988782
Longer Maturity	3.957424	1.989328	1.413311	0.974056

The MLPs has shown good performance in terms of our performance measures on all the derivatives considered. Next chapter will compare this MLPs model (henceforth referred to as MLPS II) with the closed form solutions, binomial model, LSM and LSM MLPs (henceforth referred to as MLPS I). All the models in this section considered had good performance on the in-sample dataset except of the too simple models linear regression and 2. degree polynomial regreesion. To test the predictive strength for the models, we tested the models with out-of-sample data. Specifically we consider longer maturity and deep-out-of-money options.

The MLPs is not based on a model, it is only trained on the available data. This feature makes the MLPs II pricing method versatile because the MLPs could also be used for actual market data or different models to learn patterns. The fact that the method can be extended to real market data is already shown in (Gaspar, Lopes, and Sequeira, 2020). This section showed how succesfully MLPs compared to polynomial is in a GBM model setup for pricing derivatives.

Chapter 6

Numerical Investigation and Discussion

This chapter will compare empirically all the numerical methods presented for different types of contingent claims. The first section looks at the closed form solutions to the european options compared with lattice approch for pricing.

6.1 European Options

European options are simple in the sense, that they can only be exercise at maturity. Thoughout the previous chapters specially chapter 2 and 3 we have seen closed form solutions for the european call, call on max, call on min and geometric average options. The lattice model presented is a numerical model to approximate european options and american options. The closed form solutions and the numerical lattice approch can be compared for european options, where a small deviation indicates that the lattice approach gives reasonable prices for the american options.

First we compare the numerical cost for european put min option by increasing the number of equidistant time-steps, where a trade-off between time and accuracy is investigated. The european put min has a closed form solution[1], hence the effect on accuracy is measured by comparision. The computation time is measured in order to discuss the trade-off between speed and accuracy. Tabel 6.1 shows that the algorithm accuracy increases with the number of equidistant time-steps, but the computational speed dramatically slows down for high number of steps. In MLPs II pricing method we used 100 equidistant time-steps, because we simulated 300.000 datapoints, which is a computational heavy task. We could have chosen more time-steps for a increased accuracy for the true price, but we judged the cost higher in terms of computational time than the benefits. For the other tables will we use BEG with 500 time-steps, because henceforth we need considerable less prices, hence the accuracy weights more.

Table 6.2 shows that the BEG and CRR model have highest accuracy for 100 equidistant time-steps hence we choose to price with N=100 for the multivariate binomial model. The biggest absolute difference is 0.013 which is reasonable within the bid-ask spread for most options.

From table 6.3 we see that the BEG approach The results are promising for valuation of american put options based on several underlying assets. This table will serve as reference and benchmark for the more recent approaches with neural networks in chapter 6.

[1] Equation (3.29) and (3.28)

TABLE 6.1: Comparision of speed and accuracy for a european put min option, where the inputs are K=40, $S_1(0) = S_2(0) = 40$, $\sigma_1 = 0.2, \sigma_2 = 0.3$, T=1, $\rho = 0.5$ and r=0.06. Note ms is shorthand for millisecond

Method	**No. Steps**	**Price**	**Time: min:sec.cs**
BEG	10	4.248	0:00.003
	50	4.341	0:00:097
	100	4.352	0:00.591
	200	4.358	0:04.121
	500	4.361	0:59.337
	1000	4.362	9:34.164
Analytic form		4.363	

TABLE 6.2: Valuation of multivariate contigent claims with two underlyings with K=40, $S_1(0) = S_2(0) = 40$, $\sigma_1 = 0.2, \sigma_2 = 0.3$, T=1, $\rho = 0.5$ and r=0.06.

Derivative type	**Method**	**No. Steps**	**Price**
European Call Minimum	BEG	100	2.475
	Analytic form		2.483
European Call Maximum	BEG	100	7.787
	Analytic form		7.800
Geometric Average Put	BEG	100	
	Analytic form		

6.2 American Put Option

6.3 Exotic American Options

6.4 Numerical Investigation

The computer is discrete, hence we simulate the stock path as an Bermudan option, where we have 50 time-steps per year. I.e. we approximate the American option with a Bermudan option on same underlying. We simulate 100.000 paths for the stock. By the above two algorithms for valuation, we choose to vary spot, volatility and maturity for pricing an American put option with K=40 and r=0.06. This table will serve as reference for the machine learning algorithm in chapter (!TODO chapter for machine learning). For the binomial tree we use 100 time-steps, which gives stable results (compare to figure 3.3) and for the LSM we use 10^5 paths with 50 time-steps per year. The European option is valued by using BS closed form solution for a call option (see proposition 2.16) and Put-call parity (see proposition **??**).

TABLE 6.3: Valuation of multivariate contigent claims with two underlyings with K=40, $S_1(0) = S_2(0) = 40$, $\sigma_1 = 0.2, \sigma_2 = 0.3$, T=1, $\rho = 0.5$ and r=0.06.

Derivative type	Method	No. Steps	Price
American Put Minimum	BEG	10	3.830
		50	3.884
		100	3.890

TABLE 6.4: Valuation of American put option with K=40 and r=0.06.

Spot	σ	T	Closed form European	Binomial Tree	LSM	abs. diff.
36	0.2	1	3.844	4.488	4.478	0.010
36	0.2	2	3.763	4.846	4.828	0.018
36	0.4	1	6.711	7.119	7.092	0.027
36	0.4	2	7.700	8.508	8.500	0.008
38	0.2	1	2.852	3.260	3.245	0.015
38	0.2	2	2.991	3.748	3.735	0.013
38	0.4	1	5.834	6.165	6.144	0.021
38	0.4	2	6.979	7.689	7.665	0.024
40	0.2	1	2.066	2.316	2.313	0.003
40	0.2	2	2.356	2.885	2.881	0.004
40	0.4	1	5.060	5.310	5.326	0.016
40	0.4	2	6.326	6.914	6.908	0.006
42	0.2	1	1.465	1.622	1.622	0.000
42	0.2	2	1.841	2.217	2.212	0.005
42	0.4	1	4.379	4.602	4.596	0.006
42	0.4	2	5.736	6.264	6.243	0.021
44	0.2	1	1.017	1.117	1.113	0.004
44	0.2	2	1.429	1.697	1.688	0.009
44	0.4	1	3.783	3.956	3.962	0.006
44	0.4	2	5.202	5.656	5.649	0.007

We see the maximum difference between the two algorithms is 0.027 at S=38, $\sigma = 0.4$ and T=2. The other obvious fact is that the European put has a lower value than its American counterpart, because the continuous exercise feature adds additional value to the put option.

The continuous time problem for american options is given as a optimal stopping problem:

$$\sup_{\tau \in \mathcal{T}([0,T])} E^{Q}[\exp(-r\tau)g(x_\tau)] \tag{6.1}$$

Where τ is a stopping time (definition A.5), $\mathcal{T}([0,T])$ is the class og all [0,T]-valued stopping times, g(x) is the intrinsic value of the option when the underlying state is x and the risk neutral state process assumed to be markov $\{x_t \in \mathbb{R}^d | 0 \leq t \leq T\}$. The continuous problem is often approximated by a discrete version, where the

computational procedures can be applied, hence the american option is approximated by a bermuda option. For simplicity we assume that the state process is time-homogeneous and we model a discrete pricing problem. The reformulation of the optimal stopping problem in discrete time is:

$$\sup_{\tau \in \mathcal{T}(0,\dots,T)} E^{Q}[\exp(-r\tau)g(x_\tau)]$$

We introduce the value function $J_n(x)$ which is the value of the option at time n if the underlying state process is equal to x Universal approximate theorem

The results will vary for each time running the algorithm given the stochastic nature in optimization and in sampling.

To compare the results by

TABLE 6.5: Valuation of American put option with K=40 and r=0.06.

Spot	σ	**T**	**MLPs Regression**	**Binomial Tree**	**LSM**	**abs. diff.**
36	0.2	1	4.584	4.488	4.478	0.010
36	0.2	2	4.649	4.846	4.828	0.018
36	0.4	1	7.090	7.119	7.092	0.027
36	0.4	2	8.487	8.508	8.500	0.008
38	0.2	1	3.094	3.260	3.245	0.015
38	0.2	2	3.638	3.748	3.735	0.013
38	0.4	1	6.172	6.165	6.144	0.021
38	0.4	2	7.605	7.689	7.665	0.024
40	0.2	1	2.114	2.316	2.313	0.003
40	0.2	2	2.779	2.885	2.881	0.004
40	0.4	1	5.274	5.310	5.326	0.016
40	0.4	2	6.839	6.914	6.908	0.006
42	0.2	1	1.494	1.622	1.622	0.000
42	0.2	2	2.167	2.217	2.212	0.005
42	0.4	1	4.548	4.602	4.596	0.006
42	0.4	2	6.197	6.264	6.243	0.021
44	0.2	1	1.000	1.117	1.113	0.004
44	0.2	2	1.678	1.697	1.688	0.009
44	0.4	1	3.949	3.956	3.962	0.006
44	0.4	2	5.649	5.656	5.649	0.007

(Liaw et al., 2018)

6.5 Black Scholes Model

In the B-S setup in section 2.3 we assume constant volatility, which is not realistic in terms of the known fact that volatility skew for equity options (p. 458 (Hull, 2017)).

The underlying risky equity assets is assumed in black scholes theory to evolve with a GBM, where the distribution of possible stock prices at the end of any interval is lognormal. Other models could have been considered e.g. the Bachelor model, where the differences in time for a stock are normal distributed.

$$dS_i = \sigma_i dW_i$$

This assumption would simplify the pricing problem of arithmetic basket options, because the problem is essentially one dimensionel like in the case with the geometric basket option for the black scholes model (section 3.4.1). A disadvantage with the bachelier model is it can lead to negative stock values, which is not realistic. The Black-Scholes model has some drawbacks where you can question that is the probability distribution of asset prices really lognormal.

Empirical real data from markets has not constant volatility, but the volatility depends both on the maturity and strike, hence the investors talks about volatility shew or volatility smiles (chapter 20 (Hull, 2017)). The reason why equity options have shew in the volatility term structure is that investors are more concerned with falling stock prices than rising prices, hence the volatility are instantaneously negative correlated with the stock price. To overcome the issue about assuming constant volatility a model with stochastic variance can be considered. The model becomes more complex with a extra stochastic variable where for simplicity we consider the two factor Heston model. The basic model is given by the stock follow the SDE:

$$dS(t) = \alpha S(t)dt + \sqrt{V(t)}S(t)dW_S(t)$$

And the stochastic variance process $V(t)$ is the solution to the SDE:

$$dV(t) = a(\theta - V(t))dt + \epsilon\sqrt{V(t)}dW_V(t) \quad where a > 0, \theta > 0, \epsilon > 0 \text{ and } V(0) > 0$$

Where $W_S(t)$ and $W_V(t)$ have correlation ρ. The interpretation of the constants are θ is long run average price variance, a is the rate which $V(t)$ reverts to θ and ϵ is the volatility of the volatility. The implication of ϵ is that $V(t)$ is more volatile when volatility is high and correlation between the stock and variance process reveals the negative correlation. The negative correlation between the two processes display the real market phenomenon of volatility shew for equity options, hence by assuming stochastic volatility the Heston model overcomes the issue with volatility shew.

Other models for describing the stochastic underlying process could be "Constant Elasticity of Variance model", "Merton's Mixed Jump-Diffusion Model", "Variance-Gamma Model"

Another underlying would could be the heston model
pay transaction costs and taxe
Options can be interpreted as corporate securities
credit risk and Mertons model

6.5.1 Neural Networks Vs. Linear Model

Remember that neural network does not have curse of dimensionality, hence the method compared to the linear model has advantages when considering multivariate contingent claims.

6.5.2 Subsection 2

Computational time scales exponentially for PDE methods with the number of underlyings and the same goes for the binomial lattice approach, because both methods use dynamic programming. For the monte carlo methods the computational

time only scales linear with the number of underlyings, where a smart method is needed for calculating stopping times.

Pricing options with early exercise decisions is not a issue with backward pricing methods.

6.6 Main Section 2

(Liaw et al., 2018) (Ferguson and Green, 2018)

Methods cannot be compared due to lack of code efficientcy. This is something that can be optimized.

Chapter 7

Conclusion and Further Investigation

7.1 Conclusion

7.2 Further Investigation

Appendix A

Mathematical results and definitions

Theorem A.1. ***Itô's formula multidimensional*** *Let the n-dimensional process X have dynamics given by:*

$$dX(t) = \mu(t)dt + \sigma(t)dW(t) \tag{A.1}$$

Then the process $f(t, X(t))$ has stochastic differential given by:

$$df(t,X(t)) = \frac{\partial f(t,X(t))}{\partial t}dt + \sum_{i=1}^{n}\frac{\partial f(t,X(t))}{\partial x_i}dX_i(t) + \frac{1}{2}\sum_{i,j=1}^{n}\frac{\partial^2 f(t,X(t))}{\partial x_i \partial x_j}dX_i(t)dX_j(t) \tag{A.2}$$

Note:

$$dW_i(t)\cdot dW_j(t) = \begin{cases} \rho_{ij}dt & \text{For correlated Weiner processes} \\ 0 & \text{For independent Weiner processes} \end{cases}$$

(page 58-60 (Björk, 2009))

Theorem A.2. ***The Martingale Representation Theorem*** *Let W be a k-dimensional Wiener process, and assume that the filtration is defined by:*

$$\mathcal{F}_t = \mathcal{F}_t^W \quad t \in [0,T]$$

Let M be any $\mathcal{F}_t$-adapted martingale. Then there exist uniquely determined $\mathcal{F}_t$-adapted processes $h_1, \ldots, h_k$ such that M has the representation

$$M(t) = M(0) + \sum_{i=1}^{k}\int_0^t h_i(s)dW_i(s) \quad t \in [0,T]$$

if the martingale M is square integrable, then $h_1, \ldots, h_k$ are in L^2 (page 161 (Björk, 2009)).

Theorem A.3. ***The Girsanov Theorem*** *Assume the probability space $(\Omega, \mathcal{F}, P, (\mathcal{F}_t^{\bar{W}})_{t\in[0,T]})$ and let the Girsanov kernel ϕ be any d-dimensional adapted column vector process. Choose a fixed T and define the process L on $[0,T]$ by:*
$dL_t = \phi(t)^T \cdot L_t d\bar{W}_t^P$
$L_0 = 1$.
Assume that $E^P[L_T] = 1$ and define the new probability measure Q on $\mathcal{F}_T$ by:

$$L_T = \frac{dQ}{dP} \quad \text{on } \mathcal{F}_T$$

Then

$$d\bar{W}(t) = \phi(t)dt + dW(t) \tag{A.3}$$

Where $W(t)$ is the Q-Wiener process and $\bar{W}(t)$ is the P-Wiener process (page 164 (Björk, 2009))

Theorem A.4. ***The Converse of the Girsanov Theorem*** *Let $\bar{W}$ be k-dimensional standard P-Wiener process (i.e. zero drift and unit variance independent components) on $(\Omega, \mathcal{F}, P, (\mathcal{F}_t^{\bar{W}})_{t\in[0,T]})$. Assume that there exits a probability measure Q such that $Q << P$ on $\mathcal{F}_T^{\bar{W}})$. Then there exists an adapted process ϕ such that the likelihood process L has dynamics*

$$\begin{aligned} dL(t) &= L(t)\phi^T(t)d\bar{W}(t) \\ L(0) &= 1 \end{aligned}$$

(page 168 (Björk, 2009))

Definition A.5. Stopping time: A random variable $\tau : \Omega \to [0, \infty]$ is called a Markov time if

$$(\tau \leq t) \in \mathcal{F}_t \quad \forall t \geq 0$$

A Markov time is called a stopping time if $\tau < \infty$ P-a.s. (p. 27 (Shiryaev and Peskir, 2006))

Definition A.6. Orthogonal vectors: Two vectors $\overrightarrow{a}$ and $\overrightarrow{b}$ are orthogonal, if their dot product is 0:

$$\overrightarrow{a} \cdot \overrightarrow{b} = 0$$

We will use the notation:

$$\overrightarrow{a} \perp \overrightarrow{b} \tag{A.4}$$

Definition A.7. Snell envelope Consider a fixed process Y.

- We say that a process X dominates the process Y if $X_n \geq Y_n \; P - a.s. \; \forall n$.
- Assuming that $E[Y_n] < \infty \; \forall n \leq T$, the Snell envelope S, of the process Y is defined as the smallest supermartingale dominating Y. More precisely: S is a supermartingale dominating Y, and if D is another supermartingale dominating Y, then $S_n \leq D_n \; P - a.s. \; \forall n$.

(page 380 (Björk, 2019))

Theorem A.8. ***The Snell Envelope Theorem*** *The optimal value process V is the Snell envelope of the gain process G*

Proof. see page 381 (Björk, 2019). ■

(page 381 (Björk, 2019))

Appendix B

Notation

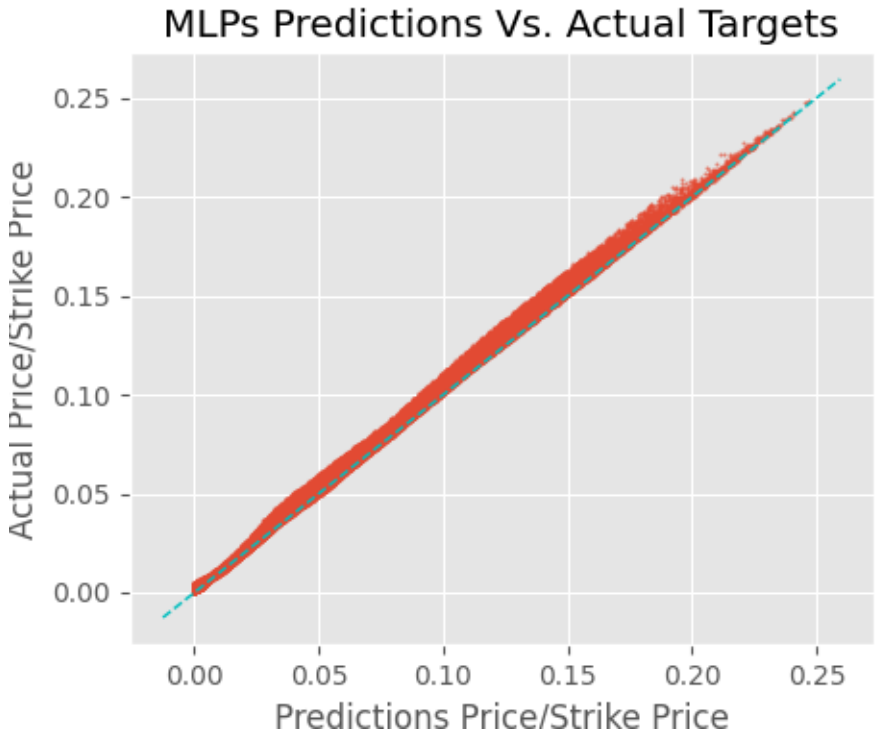

FIGURE B.1: MLPs Performance for Out-of-money datasets on American Put, where the targets are from the binomial model

We choose grid search with the cartesian product, where b denote the batchsize.

$$\eta \times b = \{(\eta, b) : \eta \in \{0.0001, 0.001, 0.01\} \; and \; b \in \{8, 256, 1024\}\}$$

From the above table we see that the best configuration on this run is for $\eta = 0.001$ and $b = 8$. LIke for the european option we will investigate if the dataset size matter for the loss of our model. We choose to the learning $\eta = 0.001$ by above test and we vary b in the set $b = \{8, 64\}$, beacuse we might suspect a higger batchsize is better for a bigger dataset.

From above table B.1 the training error improves with bigger dataset, hence we choose the dataset with 300.000 samples, where from the training set a validation set of size 60.000 is subsampled. The above study is never complete, but we have seem with grid search the optimal choice for the learning rate, batch size and datasize.

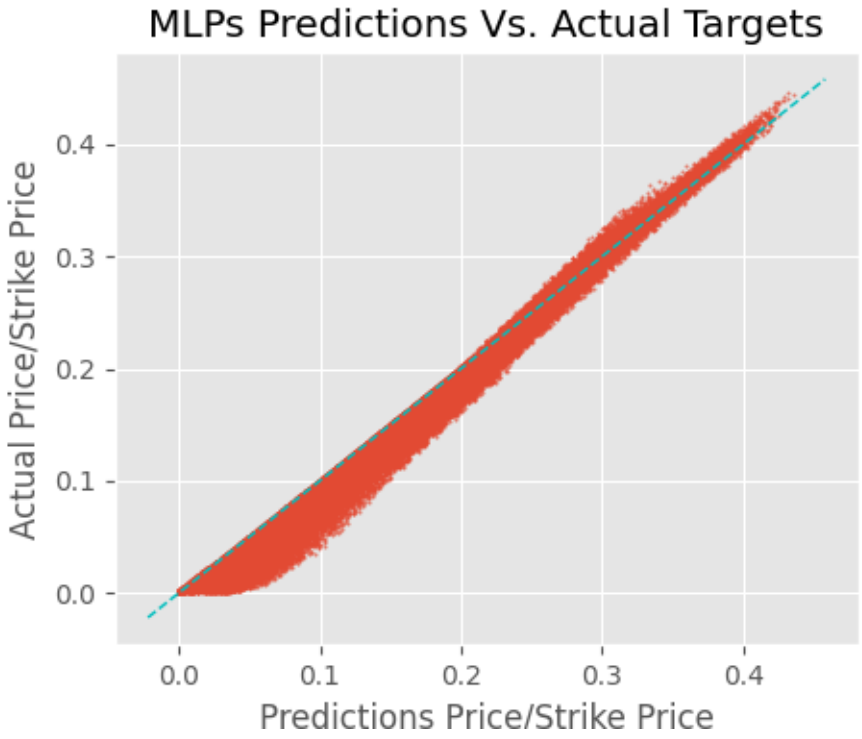

FIGURE B.2: MLPs Performance for Out-of-money datasets on American Put, where the targets are from the binomial model

Below we will try a different model than MLPs to see if we really need deep learning for this method.

Long maurity polynomial regression: ['MSE test 0.002662', 'MSE test 0.001196', 'MSE test 0.001654', 'MSE test 0.003956', 'MSE test 0.012255', 'MSE test 0.043361'] ['RMSE test 0.051593', 'RMSE test 0.034577', 'RMSE test 0.040666', 'RMSE test 0.062894', 'RMSE test 0.110702', 'RMSE test 0.208233'] ['MAE test 0.041232', 'MAE test 0.026287', 'MAE test 0.028371', 'MAE test 0.039932', 'MAE test 0.064402', 'MAE test 0.111190'] ['R2 test 0.818143', 'R2 test 0.918316', 'R2 test 0.887014', 'R2 test 0.729744', 'R2 test 0.162742', 'R2 test -1.962442']

In-sample polynomial regression: ['MSE test 0.000636', 'MSE test 0.000069', 'MSE test 0.000013', 'MSE test 0.000004', 'MSE test 0.000002', 'MSE test 0.000001'] ['RMSE test 0.025212', 'RMSE test 0.008316', 'RMSE test 0.003624', 'RMSE test 0.002052', 'RMSE test 0.001414', 'RMSE test 0.000958'] ['MAE test 0.018326', 'MAE test 0.006141', 'MAE test 0.002561', 'MAE test 0.001280', 'MAE test 0.000867', 'MAE test 0.000591'] ['R2 test 0.936628', 'R2 test 0.993105', 'R2 test 0.998691', 'R2 test 0.999580', 'R2 test 0.999801', 'R2 test 0.999909']

Out-of-money maurity polynomial regression: ['MSE test 0.005772', 'MSE test 0.000767', 'MSE test 0.000839', 'MSE test 0.000944', 'MSE test 0.001812', 'MSE test 0.004423'] ['RMSE test 0.075973', 'RMSE test 0.027694', 'RMSE test 0.028960', 'RMSE test 0.030724', 'RMSE test 0.042568', 'RMSE test 0.066506'] ['MAE test 0.060936', 'MAE test 0.022203', 'MAE test 0.020288', 'MAE test 0.020542', 'MAE test 0.027125', 'MAE test 0.041315'] ['R2 test -2.377251', 'R2 test 0.551246', 'R2 test 0.509280', 'R2 test 0.447668', 'R2 test -0.060261', 'R2 test -1.588030']

TABLE B.1: Hyperparameter tuning of learning rate and batchsize for american put minimum two assets for the interested reader see the tensorboard (link tensorboard 3)

Model	**lr**	**Batch Size**	**Loss**
MLPs	0.00010000	1024	5.7394
	0.00010000	256	2.1029
	0.00010000	8	0.92074
	0.00100000	1024	6.3428
	0.00100000	256	1.2985
	0.00100000	8	0.070993
	0.01000000	1024	6.1258
	0.01000000	256	1.4931
	0.01000000	8	8.1098

TABLE B.2: Hyperparameter tuning of dataset size and batchsize for american put minimum two assets for the interested reader see the tensorboard (link tensorboard 4)

Model	**Dataset Size**	**Batch Size**	**Loss**
MLPS	800	8	7.6495
	800	64	36.535
	80K	8	0.082552
	80K	64	0.091618
	240K	8	0.073089
	240K	64	0.091618

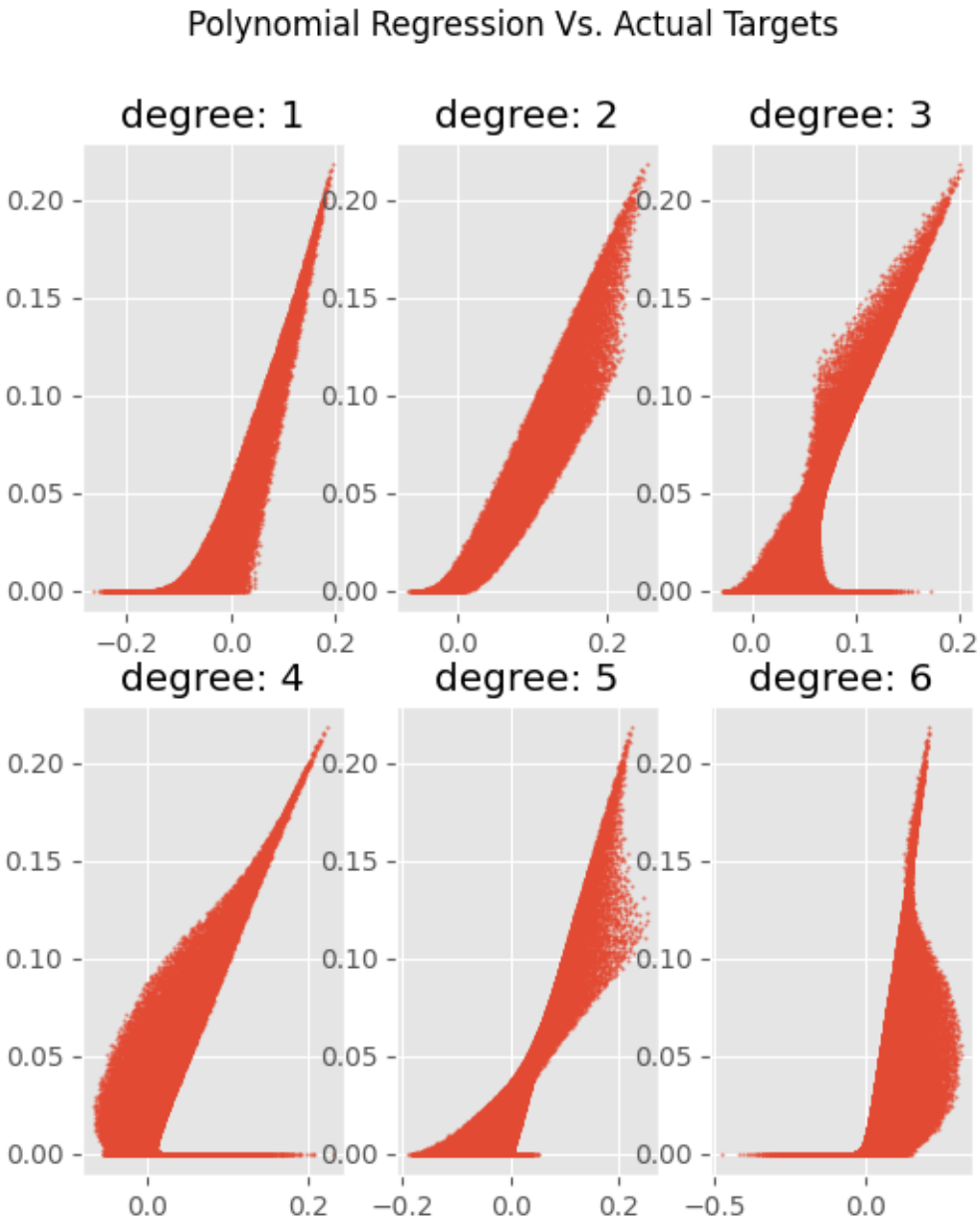

FIGURE B.3: Polynomial Regression Performance on out of money dataset on european call

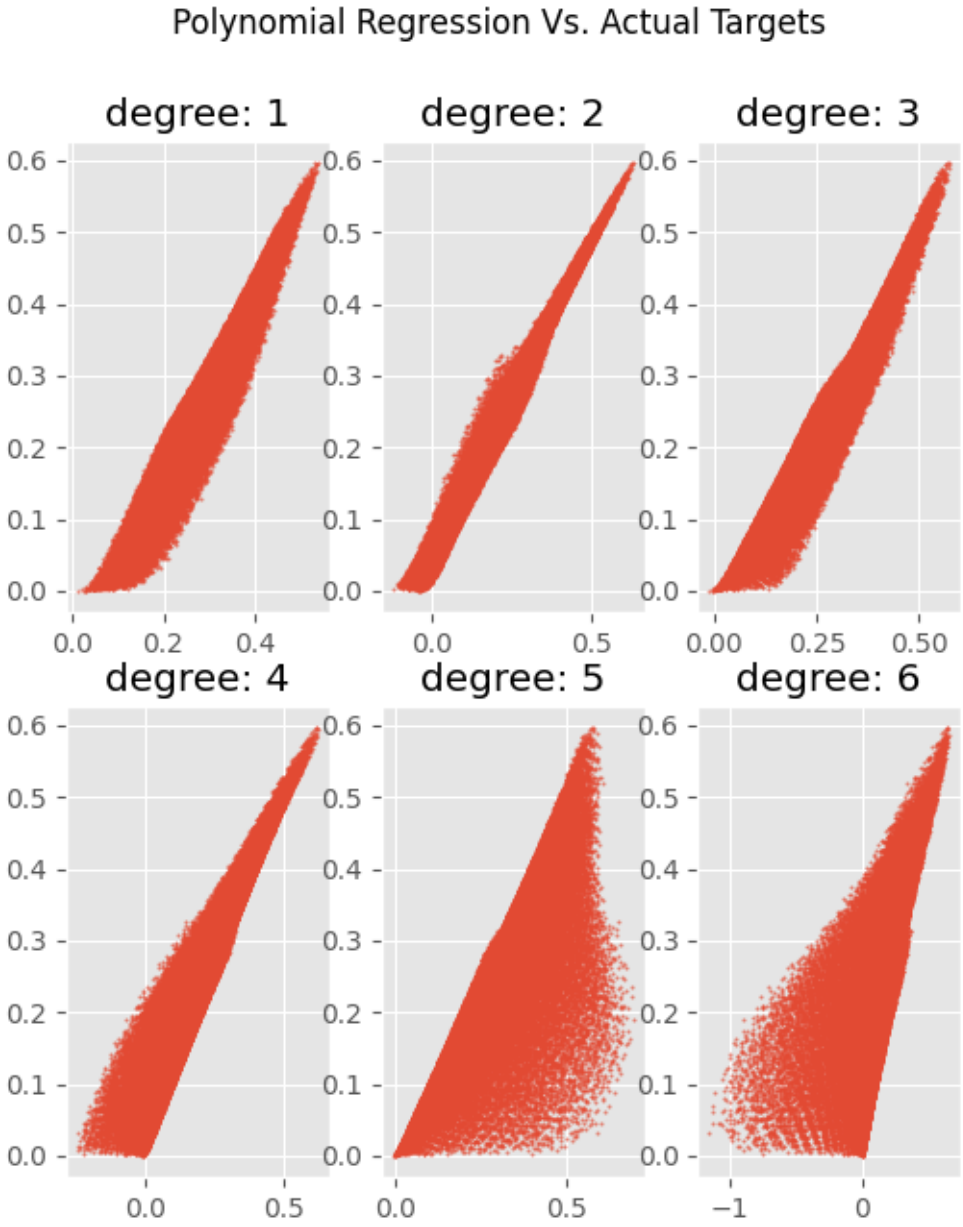

FIGURE B.4: Polynomial Regression Performance on long maturity dataset on european call

TABLE B.3: Hyperparameter tuning of dataset size and batchsize for american put minimum two assets for the interested reader see the tensorboard

Model	**Dataset Size**	**Batch Size**	**Loss**
300K	0.0001	8	0.015130897983909
300K	0.001	64	0.035523075610399
100K	0.0001	8	0.064886227250099
100K	0.001	8	0.072143875062466
300K	0.0001	64	0.075988814234734
300K	0.001	8	0.104622706770897
300K	0.01	256	0.480043411254883
300K	0.0001	256	1.04093873500824
300K	0.001	256	1.06809389591217
300K	0.001	512	1.08942472934723
100K	0.001	64	1.18230485916138
300K	0.001	1024	1.30065310001373
300K	0.01	512	1.45135951042175
100K	0.01	256	1.52623510360718
300K	0.01	64	1.75954759120941
300K	0.01	1024	1.92213356494904
100K	0.01	512	2.01193928718567
100K	0.01	64	2.02969717979431
300K	0.0001	1024	5.72439670562744
300K	0.0001	512	5.72847843170166
100K	0.0001	256	5.76278400421143
100K	0.0001	512	5.77397203445435
100K	0.0001	64	5.80239868164063
100K	0.0001	1024	5.90070772171021
100K	0.001	256	6.29003953933716
100K	0.001	512	6.52896738052368
100K	0.001	1024	6.56187915802002
1K	0.001	8	7.5845251083374
100K	0.01	8	8.07239627838135
1K	0.01	8	11.7414264678955
100K	0.01	1024	13.4287099838257
1K	0.01	64	14.1254253387451
1K	0.0001	8	24.0087566375732
1K	0.001	64	27.0112972259521
1K	0.01	512	40.0226554870605
1K	0.001	512	44.841724395752
1K	0.01	256	45.8363571166992
1K	0.01	1024	49.3933792114258
1K	0.0001	64	51.3070755004883
1K	0.001	256	52.8623466491699
1K	0.0001	512	87.4313125610352
1K	0.0001	256	89.8093795776367
1K	0.0001	1024	90.8342361450195

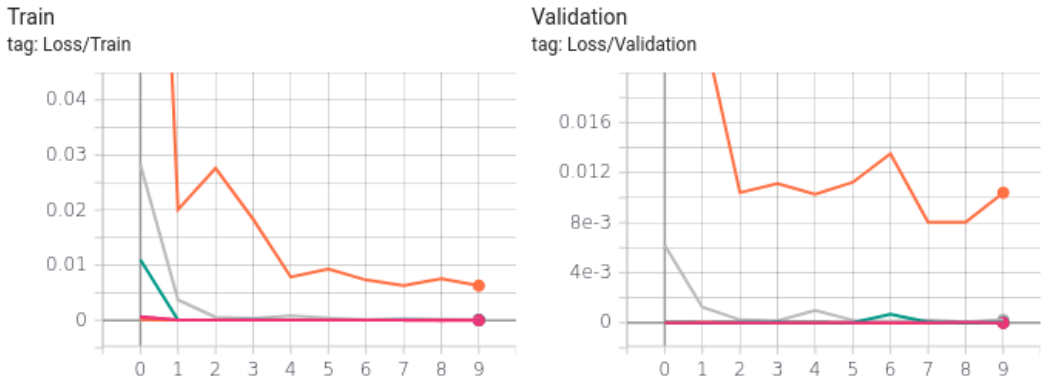

FIGURE B.5: MLPs regression training for european call, where the plot shows the effect of gather more data on training and validation loss, where the x-axis is number of epoch-1 and y-axis is the loss for the training and validation loop. The orange line with the largest error is for 100 datapoints, the grey line for 1000 datapoints, the green line 10K datapoints, the three last lines blends together where it is for 100K,300K and 1M datapoints. More can be seem at tensorboard (link tensorboard 5)

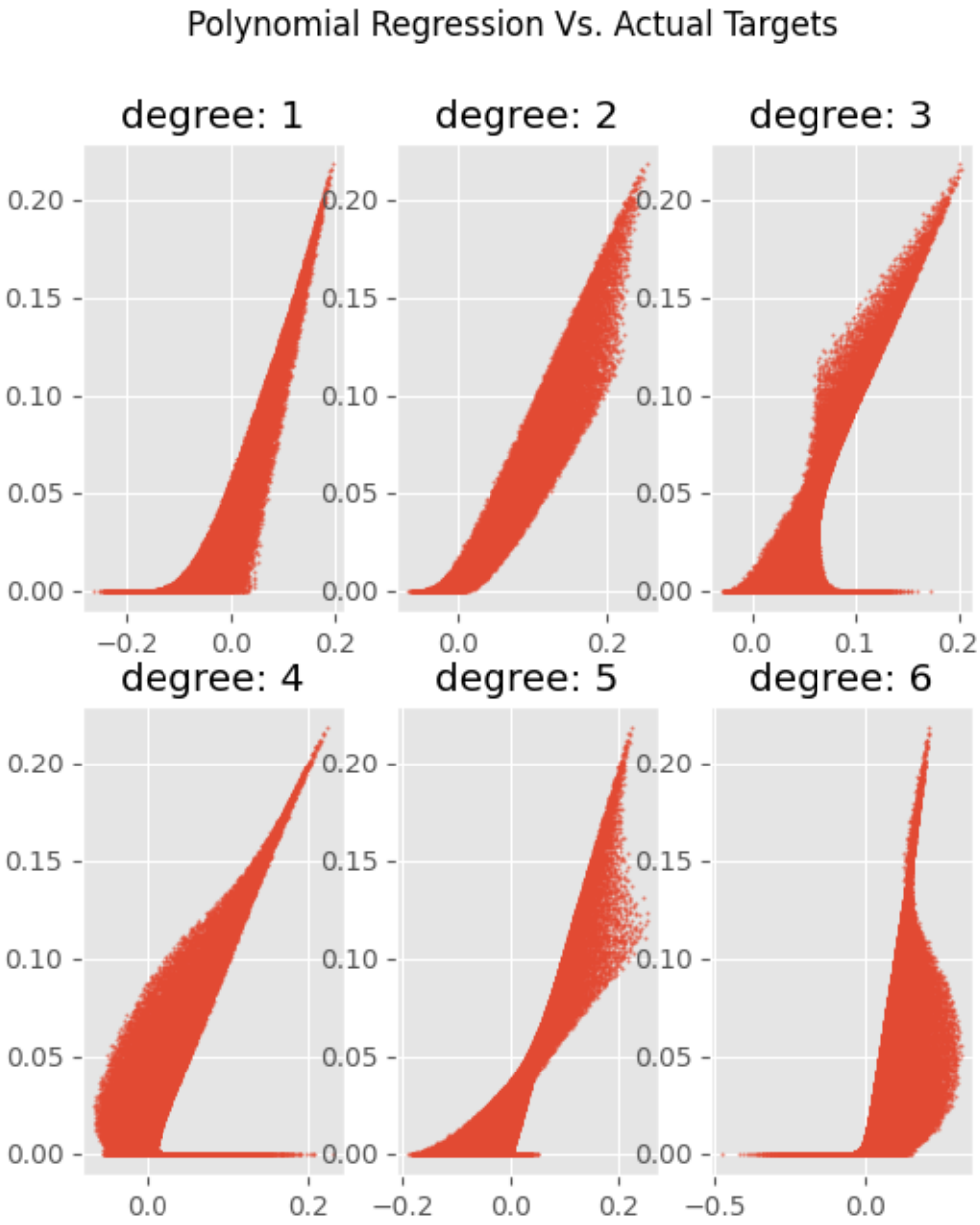

FIGURE B.6: Predicted price based on polynomial regression of varying degree

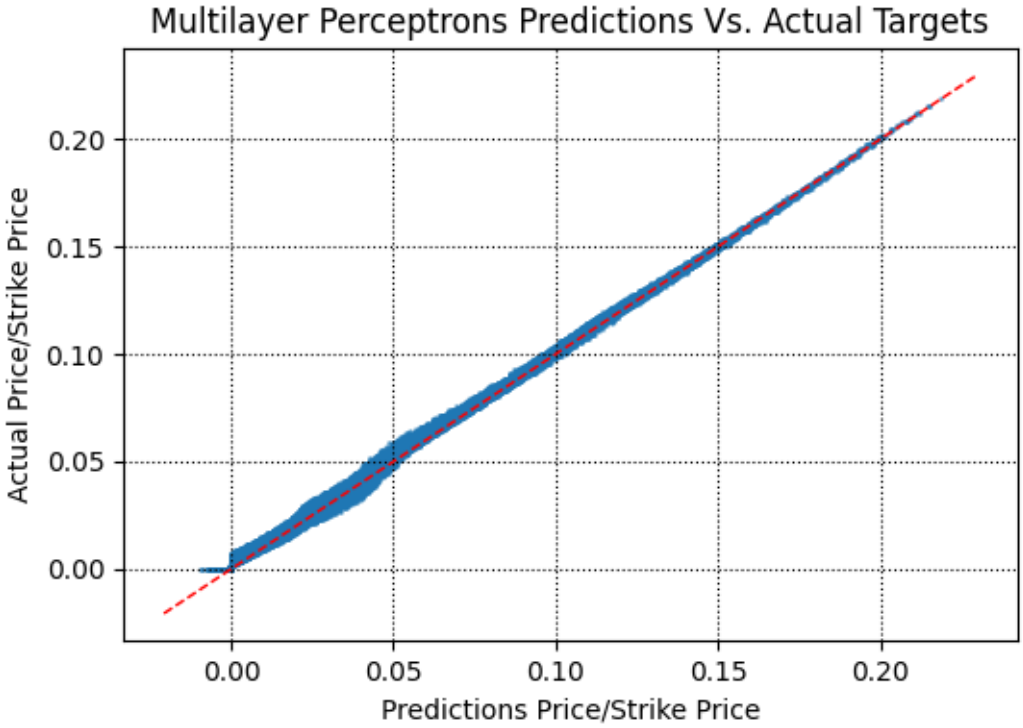

FIGURE B.7: Predicted price based on MLPs model

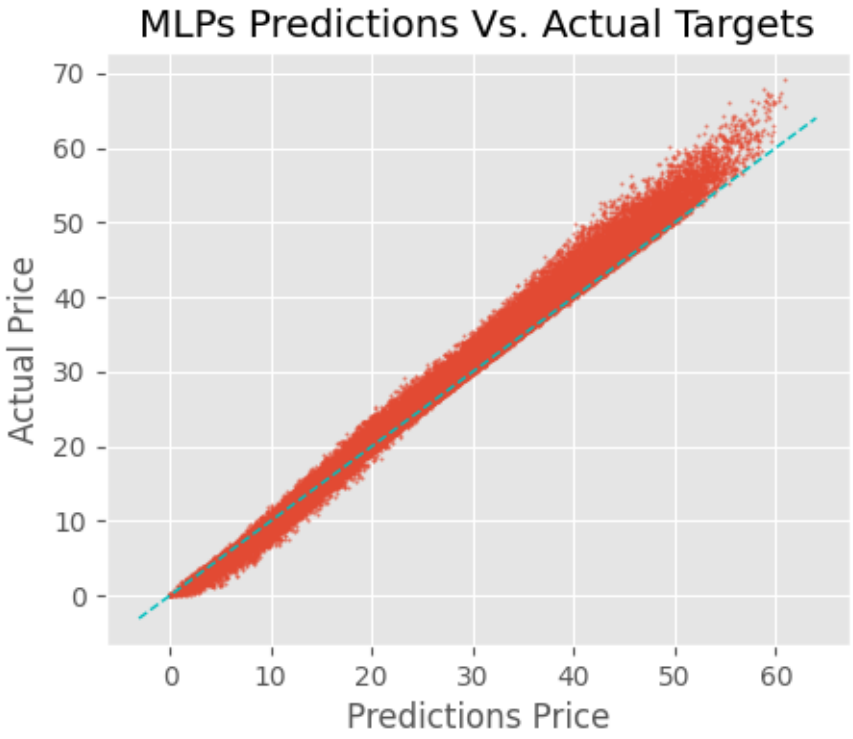

FIGURE B.8: MLPs Performance on long maturity dataset on American put on minimum of two stocks

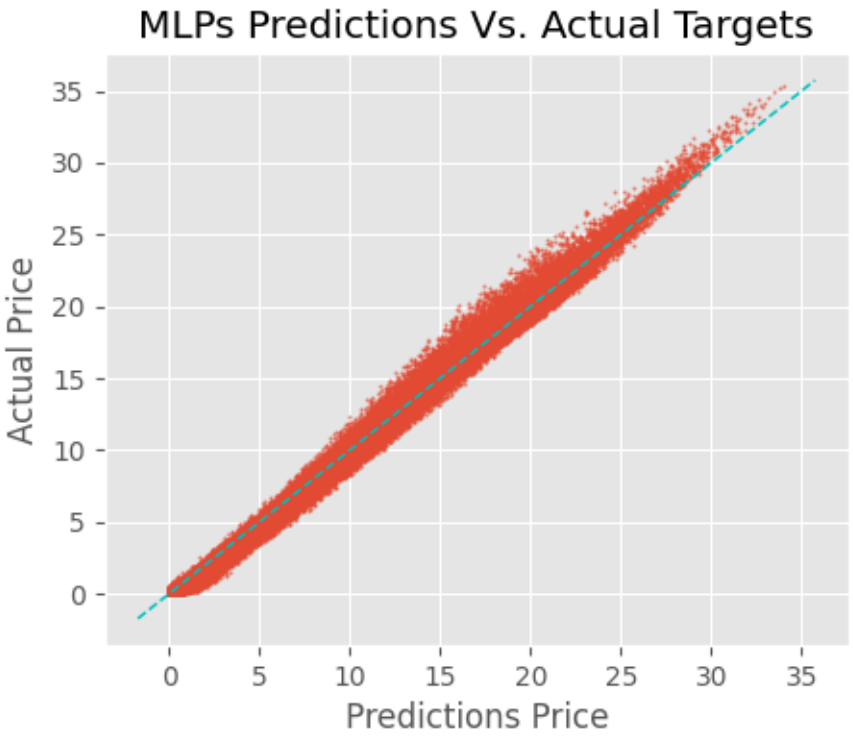

FIGURE B.9: MLPs Performance on long maturity dataset on American put on minimum of two stocks

Bibliography

Andersen, Leif and Mark Broadie (2004). "Primal-Dual Simulation Algorithm for Pricing Multidimensional American Options". eng. In: *Management science*. Management Science 50.9, pp. 1222–1234. ISSN: 1526-5501.

Björk, Thomas (2009). *Arbitrage Theory in Continuous Time*. Third edition. Oxford.

Björk, Tomas (2019). *Arbitrage Theory in Continuous Time*. eng. Oxford Finance Series. Oxford: Oxford University Press. ISBN: 9780198851615.

Black, Fischer and Myron Scholes (1973). "The Pricing of Options and Corporate Liabilities". In: *The Journal of Political Economy* 81.3, pp. 637–654. URL: http://www.jstor.org/stable/1831029 (visited on 02/09/2020).

Boyle, Phelim P., Jeremy Evnine, and Stephen Gibbs (1989). "Numerical Evaluation of Multivariate Contingent Claims". eng. In: *The Review of financial studies* 2.2, pp. 241–250. ISSN: 0893-9454.

Buffett, Warren (2002). "Berkshire Hathaway's (BRK.B) annual letters to shareholders". In: URL: https://www.berkshirehathaway.com/letters/2002pdf.pdf (visited on 06/23/2020).

Clément, Emmanuelle, D. Lamberton, and Philip Protter (Sept. 2001). "An analysis of the Longstaff-Schwartz algorithm for American option pricing". In:

Cox, John and Mark Rubinstein Stephen Ross (1979). "Option princing: A simplified approach". In: *Journal of Financial Economics* 7, pp. 229–263.

Ekvall, Niklas (1996). "A lattice approach for pricing of multivariate contingent claims". In: *European Journal of Operational Research* 91.2, pp. 214–228. URL: https://EconPapers.repec.org/RePEc:eee:ejores:v:91:y:1996:i:2:p:214-228.

Elliott, Robert J. and P. Ekkehard Kopp (1999). *Mathematics of financial markets*. eng. Springer finance. Berlin: Springer. ISBN: 0387985530.

Ferguson, Ryan and Andrew Green (2018). "Deeply Learning Derivatives". eng. In:

Gaspar, Raquel M, Sara D Lopes, and Bernardo Sequeira (2020). "Neural Network Pricing of American Put Options". eng. In: *Risks (Basel)* 8.3, pp. 73–. ISSN: 2227-9091.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Hirsa, Ali, Tugce Karatas, and Amir Oskoui (2019). "Supervised Deep Neural Networks (DNNs) for Pricing/Calibration of Vanilla/Exotic Options Under Various Different Processes". eng. In:

Hull, John C. (2017). *Options, Futures, and Other Derivatives*. Vol. Ninth edition. Pearson Global Education.

Johnson, Herb (1987). "Options on the Maximum or the Minimum of Several Assets". In: *The Journal of Financial and Quantitative Analysis* 22.3, pp. 277–283. URL: https://www.jstor.org/stable/2330963?seq=1.

Kohler, Michael, Adam Krzyżak, and Nebojsa Todorovic (2010). "PRICING OF HIGH-DIMENSIONAL AMERICAN OPTIONS BY NEURAL NETWORKS". eng. In: *Mathematical finance* 20.3, pp. 383–410. ISSN: 0960-1627.

Lapeyre, Bernard and Jérôme Lelong (2019). "Neural network regression for Bermudan option pricing". eng. In:

Liaw, Richard et al. (2018). "Tune: A Research Platform for Distributed Model Selection and Training". In: *arXiv preprint arXiv:1807.05118*.

Longstaff, Francis A. and Eduardo S. Schwartz (2001). "Valuing American Options by Simulation: A Simple Least-Squares Approach". In: *The Review of Financial Studies*.

MacKay, David J. C. (2018). *Information theory, inference and learning algorithms*. eng. Repr. with corr. Cambridge University Press. ISBN: 0521642981.

McDonald, Robert and Anna Paulson (2015). "AIG in Hindsight". eng. In: *The Journal of economic perspectives* 29.2, pp. 81–106. ISSN: 0895-3309.

Merton, Robert C. (1973). "Theory of Rational Option Pricing". In: *The Bell Journal of Economics and Management Science* 4.1, pp. 141–183. URL: http://links.jstor.org/sici?sici=0005-8556%28197321%294%3A1%3C141%3ATOROP%3E2.0.CO%3B2-0 (visited on 05/14/2020).

Ouwehand, P. (2006). "PRICING RAINBOW OPTIONS". In:

OVERHAUS, MARCUS et al. (2007). *Equity Hybrid Derivatives*. eng. 1st ed. Wiley finance series. Hoboken, N.J: Wiley. ISBN: 0471770582.

Shiryaev, Albert and Goran Peskir (2006). *Optimal Stopping and Free-Boundary Problems*. eng. Lectures in Mathematics. ETH Zurich. Basel: Springer Basel AG. ISBN: 3764324198.

Tsitsiklis, J.N and B van Roy (1999). "Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives". eng. In: *IEEE Transactions on Automatic Control* 44.10, pp. 1840–1851. ISSN: 0018-9286.